User guide

PowerMACS 4000

Atlas Copco Tools and Assembly Systems

9836 3521 01
2009-11
Edition 10.0.0

Atlas Copco

Contents

Contents

Contents

Contents

Contents

1 Introduction

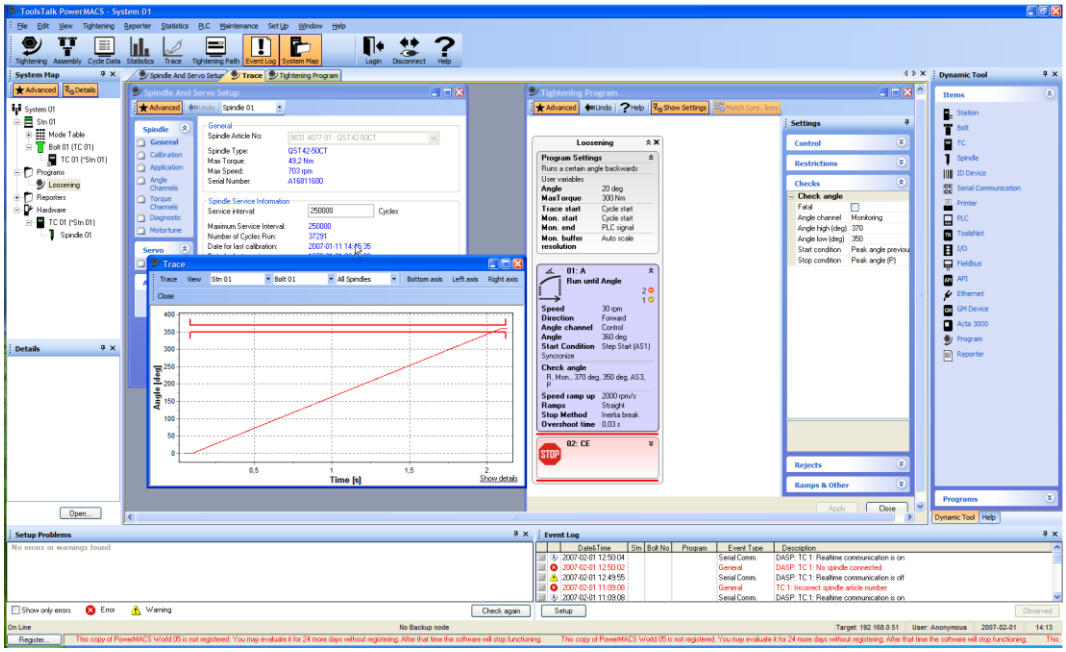
1.1 Introduction - Overview

The PowerMACS 4000 system is the seventh generation of tightening systems from Atlas Copco Tools and Assembly Systems. It is the perfect solution for all tightening applications in your assembly plant, however complex.

Building on the successfull PowerMACS concept the PowerMACS 4000 delivers even more performance and benefits to customers by increased intelligence in tightening controllers and simplified configuration through the new ToolsTalk PowerMACS software.

PowerMACS 4000 supports all known tightening and process monitoring methods and builds further on the proven PowerMACS concept to deliver even more performance using the new intelligent QST spindles.

PowerMACS 4000 is not only the most productive, cost-effective and user-friendly tightening system on the market today, it also offers an impressive new level of installation flexibility.



1.1.1 ToolsTalk PowerMACS, the friendly user interface

The new improved ToolsTalk PowerMACS (TTPM) builds on the experiences from the earlier WinTC and with increased simplicity and flexibility make the user experience more rewarding without loosing any functionality.

Simplicity has been greatly increased by the introduction of the **Basic** and **Advanced** concepts into ToolsTalk PowerMACS, most functionality is hidden for inexperienced users but available at a mouse-click for more those who feel they need the extra functionality.

Increased user friendliness by drag & drop functionality.

Full reporting ToolsTalk PowerMACS offers full reporting of tightening results, set values and limits. An easily adapted annunciator shows the tightening results in read-at-a-glance graphics. The combined SystemMap shows the tightening status of the bolts and the status of all equipped hardware. The system provides reports for any chosen parameters, trace curves, and advanced SPC and log files.

User guides Accessible via Hyperlinks, the software includes SetUp Wizards, Tutorials, Tightening Templates which guide you through the most common tightening sequences, Spare Parts Lists with drawings and article numbers, service, fault-finding and FAQ functions, and a complete on-line manual.

If you need to configure the system for Ethernet or Fieldbus communication, the PowerMACS 4000 Set Up Wizards make the task simple.

API for easy access ToolsTalk PowerMACS includes the popular Application Program Interface (API) enabling the engineer to access data from the Tightening Controller. For example: read and write PLC, cycle and trace data and store cycle data to media. All internal and external communications are via Ethernet TCP/IP thin wire 100 Mbit/s.

PowerMACS 4000 can be tailored to any assembly process. A built-in PLC (IEC 61131-3), fully programmable using the ToolsTalk interface, supports various in-station control requirements.

1.1.2 Intelligent tightening controllers, TCs

PowerMACS 4000 is modular in concept, utilizing the new improved intelligent Tightening Controller (TC) with state-of-the-art software and a wide range of peripheral support and communication possibilities. The TC is designed for IP43 and is easy to integrate into a production line as a stand-alone unit or integrated in a panel.

Even more expanded capacity than it's predecessor it is an attractive piece of industrial design, the PowerMACS 4000 TC is equipped with an impressive amount of memory, more than enough to handle the Real- Time Operative System, Application Programs and the PLC program.

Tightening controllers are now separated in the Primary Tightening Controller (TC) featuring more connectability and external communication possibilities and the more basic tightening controller (TC).

One Primary TC (PTC) is configured as a System TC and the rest as Spindle TCs. In a system all Spindle TCs can be replaced without needing to download the setup once again. One system can include a maximum of 15 stations. Up to 50 TC's can be connected in one system, controlling 50 nut runners.

Full interchangeability means fewer spare parts are required, maintenance time is reduced and availability is increased.

All common interfaces All PTC's are designed to interface with Ethernet, the most common Fieldbuses, and industrial PLCs. The Application Program Interface (API) ensures easy external access to the system.

Peripherals simply added A wide range of peripherals can be connected to the PowerMACS 4000 TC including printers, bar code readers, operator table, operator annunciators, etc.

Flexible installation The plug-and-play modular design of the TC means maximum installation flexibility. Series communication in line control and data collection enables the PowerMACS TC's to be discreetly integrated into the production line by the connection of two cables. The appropriate reporting system and suitable operator interface can be mounted next to the work area.

1.2 Tightening and Gauging

There are two types of PM4000 products:

- PM4000 Tightening
- PM4000 Gauging

PM4000 Tightening contains all functions needed for normal tightening of bolts. It can read and write Tightening setups on disk or a TC. It cannot read or write a Gauging setup.

PM4000 Gauging has all the functions of PM4000 Tightening but also additional functions for gauging applications. It can read a Tightening setup from file and convert it to a Gauging setup but not vice versa.

1.2.1 Gauging Functions

The following functions are only avaliable if you run PM4000 Gauging:

- Step BCT
- Step Run to Position
- Step Wait, PLC signal addition
- Check Delta T
- Check Low Spot
- Check Pos
- Check Peak T, additions
- Zones (within step)
- Store positions
- Backlash compensation

1.2.2 Gauging Applications

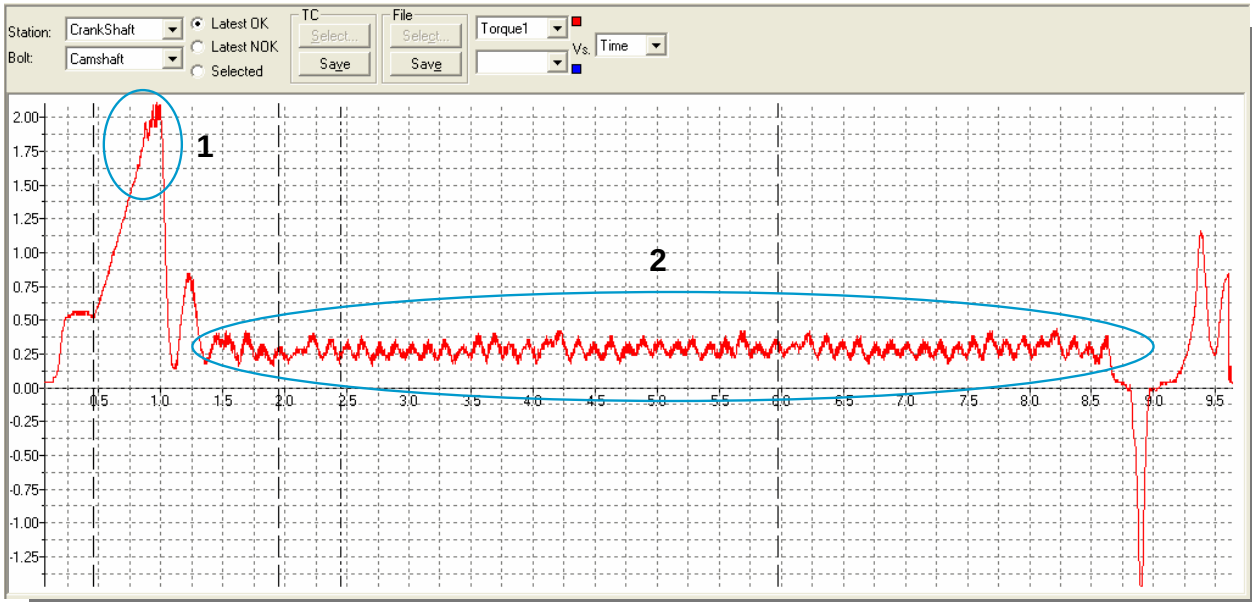
PM4000 Gauging can be used for applications where turning and measuring of torque and angle with high precision is needed. Examples:

- Torque To Turn
- Valve lash
- Indexer
- Hub nuts

1.2.2.1 Torque To Turn Gauging Application

An example of a Torque To Turn application is when an engine manufacturer wants to rotate the crank or cam shaft under controlled conditions for the first time. Metal chips, burrs, sharp edges (spot distortions) may cause excessive drag and aventually lead to early bearing shell failure and warranty repairs.

With PM4000 Gauging it is easy to create a program that rotates the shaft 360 or 720 degrees and continuously monitor the torque and angle.



The curve present torque vs. time. There are two critical points.

1. Break-Away
2. Low Torque Rotation Test

To check break-away the torque is gently increased without rotation. If the torque reaches a preset limit before any rotation starts the test stops immediately, preventing any damage.

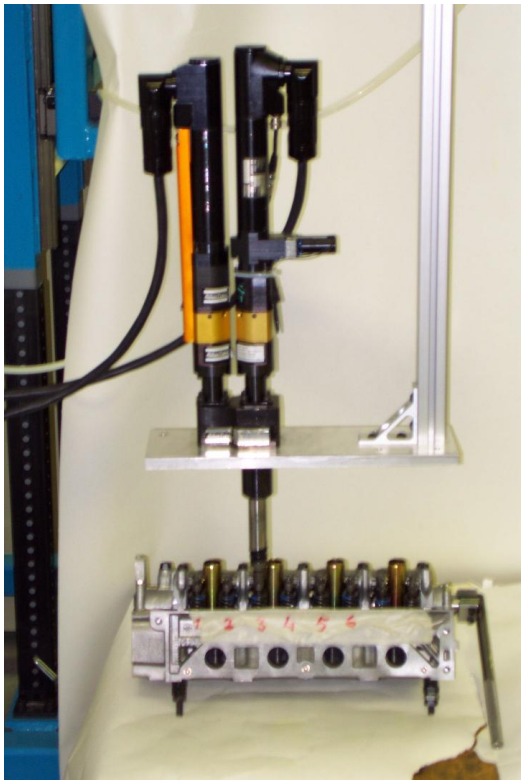
If the break-away torque is within limits the rotation starts and the torque is monitored for the full rotation at slow speed. Any friction out of limit indicates mechanical problems that should be corrected.

The combination of the two tests will show up any mechanical problems at a very early stage, before the motor has been built into e.g. a vehicle. This eliminates the risk that the motor later experience expensive service warranties.

1.2.2.2 Valve Lash Gauging Application

In a Valve Lash application a bolt and a nut should be rotated and locked to each other to achieve a specific valve lash. This can be done with a special coaxial spindle, e.g. QMX 42-2RT.

It is easy to create a program that operates the two spindles in a controlled way to make up perfectly adjusted valve lash.

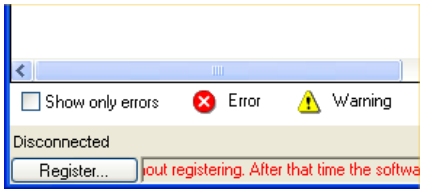


1.3 Installing ToolsTalk PowerMACS

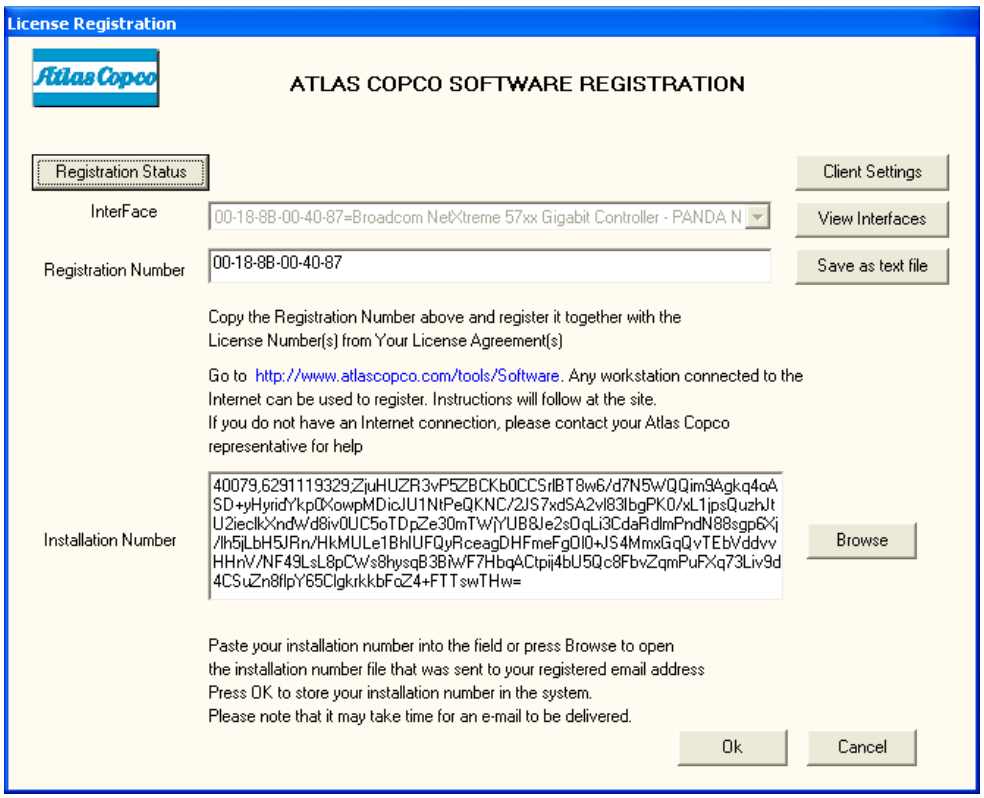
You can install ToolsTalk PowerMACS from CD or from a hard disk. To install PM4000 Tightening you excecute the file “PM4000 10.X.X ToolsTalk.exe” located in the folder ToolsTalk. PM4000 Gauging is installed with the file “PM4000 10.X.X ToolsTalk Gauging.exe” located in the folder ToolsTalk Gauging.

After installing ToolsTalk PowerMACS you have up to 90 days to register it.

Register buttons are found at the bottom left part of the window for an unregistered ToolsTalk, and in the **Help-About** window.



To register, press a Register button to invoke the below form, and follow the instructions.

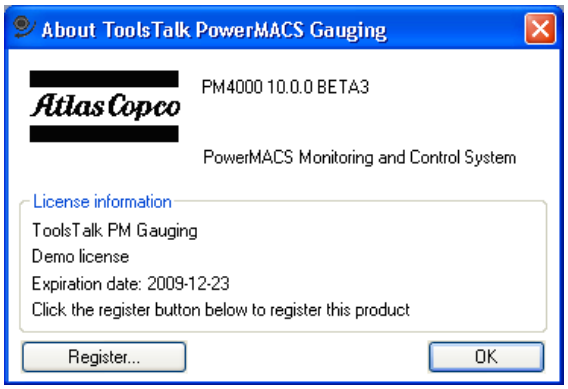


When registering please note that there are two types of products:

- PM4000 Tightening
- PM4000 Gauging

The license only covers one of the products. For more information on differences between these products see Tightening and Gauging.

Until registered ToolsTalk PowerMACS will display an extra status bar at the bottom of the main window showing you the number of days left of the evaluation period. When this time expires the application will no longer be possible to start and is automatically shut down if it is running.



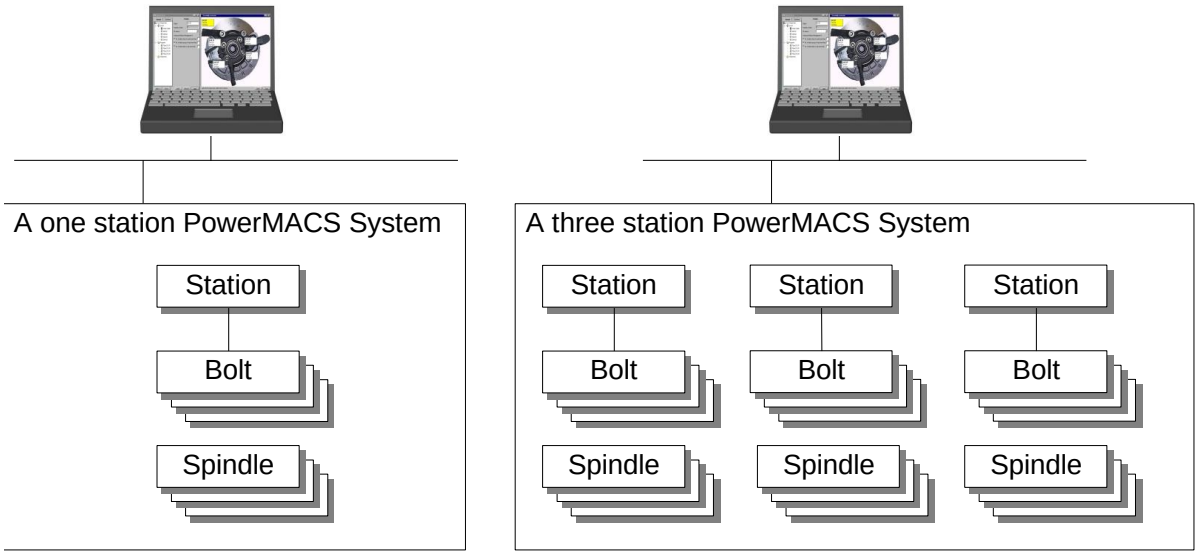
2 System Architecture

2.1 System Architecture - Overview

This section describes the architecture of the PowerMACS 4000 system. It includes the descriptions of the hardware components, including computers and devices, the structures of the system, station and spindles.

2.2 System Structure

A PowerMACS System consists of one or more stations where each station may control one or more bolts.



System

A *system* comprises one or more Tightening Controllers linked together. A system works autonomously, performing the programmed tightening task(s).

ToolsTalk PowerMACS can only connect to one PowerMACS system at the time. This means that the user cannot edit or display data from two or more systems simultaneously while using one ToolsTalk. If you need to display data from two or more stations at the same time you must design them in one system.

Multiple PowerMACS systems may be networked together in a single facility but are limited by the number of available IP addresses. There is no limitation when networking is not used.

One system can contain up to 50 Tightening Controllers and up to 50 Spindles.

Station

Within a system you can have up to 15 *stations* and 100 *Bolts* totally. One station can control up to 50 bolts but there must be no more than 100 Bolts in a system. The Station has a PLC to control the tightening cycles and interface with automation controllers. Cycle data is generated for each Station consisting of the result of all bolts controlled by the Station.

Bolt

A *Bolt* is the object that will be tightened by the system. A spindle does the actual tightening.

Spindle

A *Spindle* performs the actual tightening task. One Spindle can be used to tighten several Bolts and it normally comprises of:

- Motor
- Intelligent CPU with complete measuring system
- One or more torque sensors
- One or more angle sensors

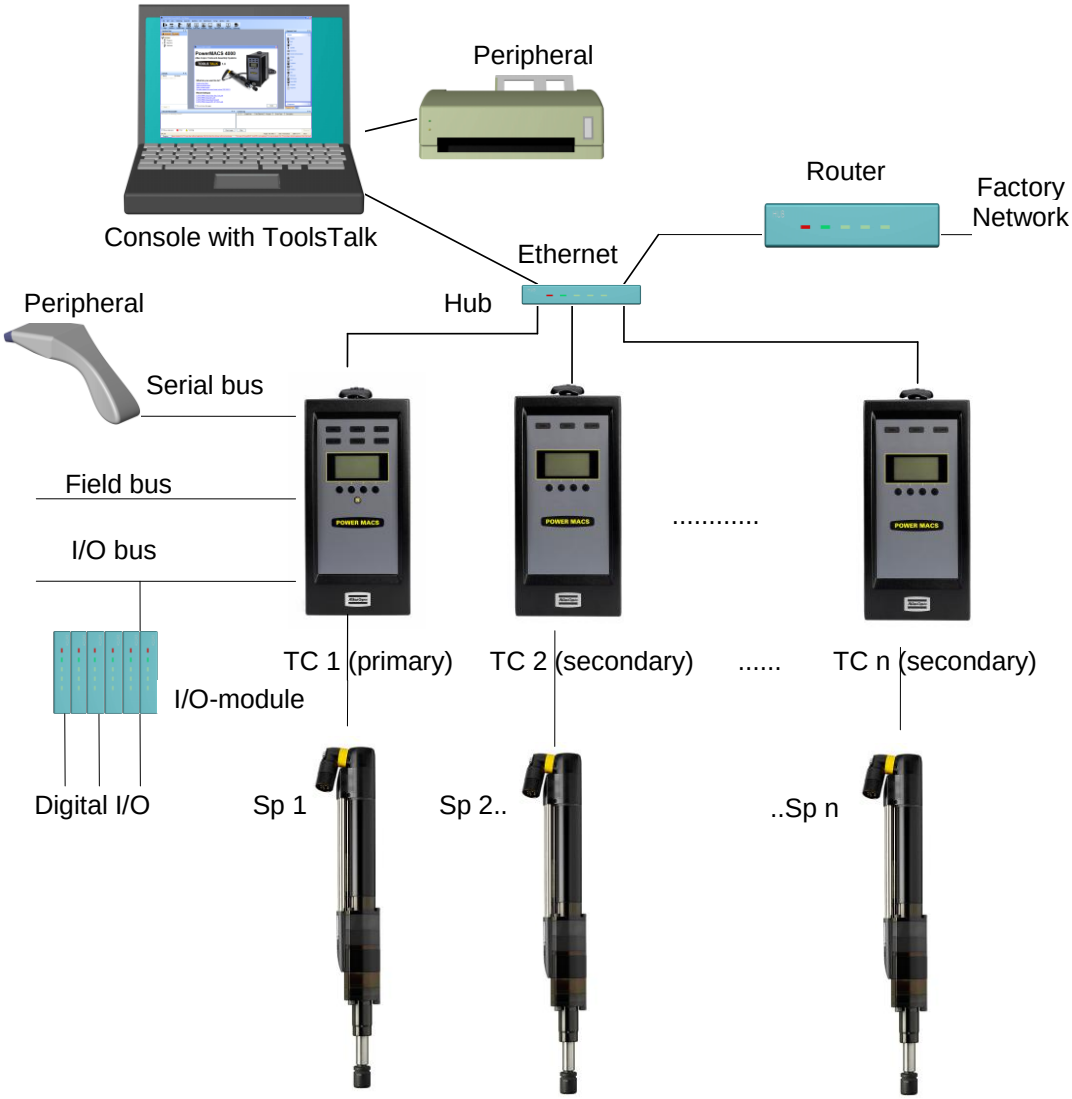
ToolsTalk

ToolsTalk PowerMACS is a Windows application used to set up a PowerMACS 4000 system. It is not needed for automatic running, but it can optionally be used to monitor the system and to collect and display various data.

ToolsTalk PowerMACS can be used for multiple systems, if they are hooked up on the same computer network, but only one system at a time can be accessed.

2.3 Hardware Structure

The figure below describes the *hardware structure* of a PowerMACS 4000 system, i.e. how hardware components are used.



2.3.1 Tightening Controller

A Tightening Controller (TC) takes care of various tasks around tightening: driving and controlling a spindle, measuring and storing angle and torque, running PLC, etc. Each TC contains all functions necessary to monitor and control one (1) spindle.

Two different Tightening Controllers exist, the **PTC** is intended to run as a System TC and S*tation TC*. The **PTC** has more external interfaces (sockets) than the standard **TC**.

The System TC is the TC that communicates directly with ToolsTalk PowerMACS. It is on this TC that the setup, downloaded from ToolsTalk PowerMACS, is stored and is responsible for starting the system when powered on, or after downloading a new setup.

A single TC belongs to one system only. The System TC, and the setup downloaded to it, defines the system. When the System TC starts up, either at power on or after receiving a new setup, it will try to attach to the Spindle TCs it needs. It will assume that IP-address of the Spindle TCs are consecutive to its own IP-address. That is, if the System TC has address 192.168.0.1 TC 2 should have 192.168.0.2, TC 3 192.168.0.3, and so on.

A Spindle TC will only accept the request of the first System TC that attach to it after a cold start. It will then remember the IP-address of the System TC and refuse requests from a System TC having another IP-address until cold started again. See chapter: Configure Target System for more information on how to configure a TC as a System TC and set its IP-address. Chapter: Check for System Conflicts describes a tool for detecting overlapping systems, that is, several System TCs trying to use the same Spindle TC.

Spindle TCs can be replaced while the system is operational, without requiring the setup to be downloaded to the new TC before starting it again. They will fetch their part of the setup from the System unit when activated.

For security and redundancy a back up copy of the setup is maintained on the second TC in a PowerMACS 4000 system. This means that the System TC in a system can be replaced without requiring the setup to be downloaded to the new System TC before starting the system again. See chapter: Handling of the setup in the target system for details.

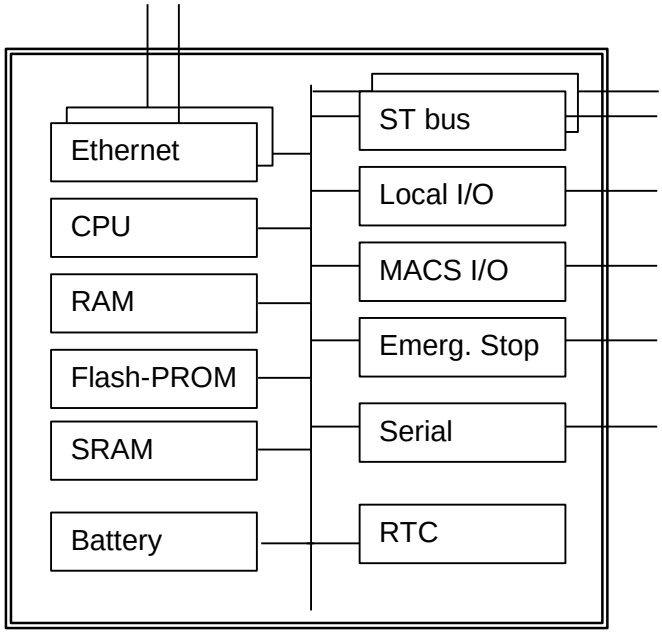
Should both the System TC and TC 2 be found nonfunctional and must be replaced then the setup must be downloaded from a backup media using ToolsTalk PowerMACS. You should therefore always keep backups of your setups in a safe place.

Note! After replacing the System TC you are strongly recommended to cold start the complete system.

All TCs in a system, System TCs as well as Spindle TCs, **must** use the same TC System Software version in order to function properly. There are several methods to verify that this is the case for all TCs in a system. The TCs themselves checks for version conflicts at start up and indicates this using the display on the their front (see chapter: Version conflict message) and ToolsTalk PowerMACS has several forms that can be used for the same purpose (see chapter: Configure Target System and Check for System Conflicts).

System Architecture

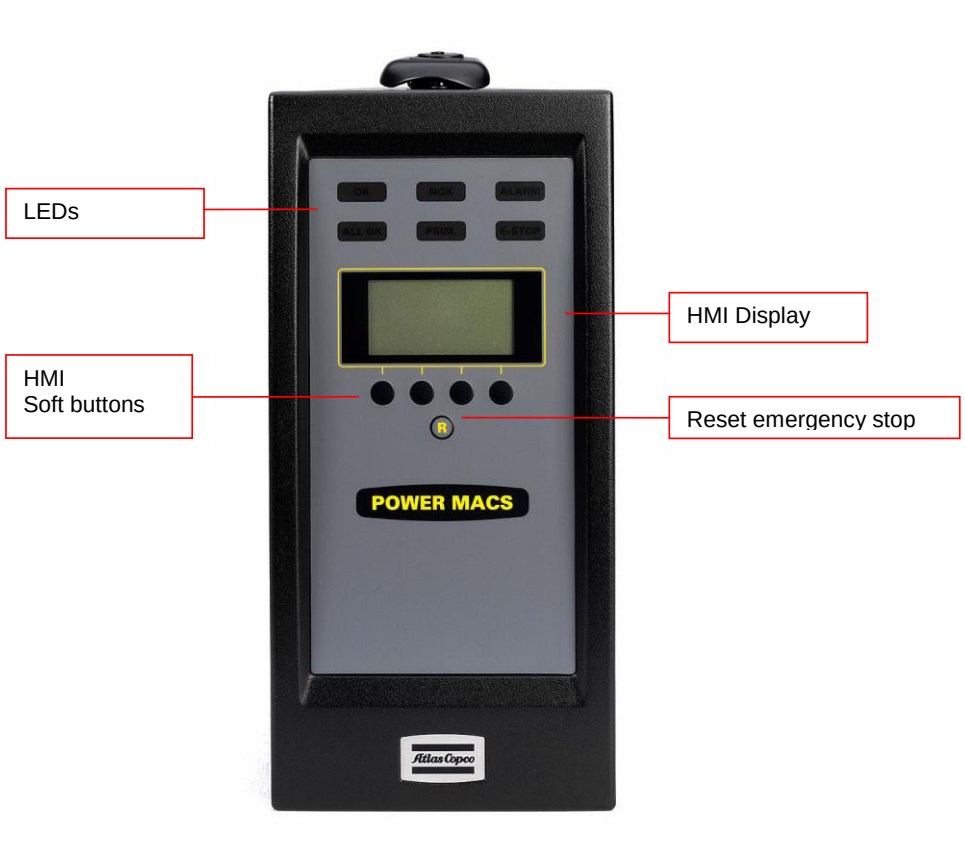
The TC board has the following layout:



Item	Function	Use
Ethernet	Ethernet interfaces	For communication with other TCs. In System TC:s the second Ethernet interface is used for communication with the Consol Computer (ToolsTalk PowerMACS), 100 MBits
CPU	Central Processing Unit	PowerPC 880, 120 MHz
RAM	Random Access Memory	For storage of program and data when executing, 32 MB
Flash-PROM	Programmable Read Only Memory of Flash type	For storage of boot and application software.
SRAM	Static RAM with battery backup	For storage of set-ups, traces, cycle data etc. 2 MB
ST bus	St bus (2 ch)	ST bus 1 is for communication with the drive and the spindle, located in the tool connector. ST bus 2 is for communication with ST based accessories.
Emerg. stop	Emergency Stop	Class 3 Emergency Stop, breaks the gate drive voltage.
Local DI	Digital input	Opto Isolated Digital Input
Local DO	Digital Output	Relay Outputs
Serial	Serial interface	Interface to serial devices, 3 channels RS232, 1 channel RS422, 2 channels RS485
MACS I/O	CAN bus interface	Opto Isolated interface to MACS I/O and other accesories
AnyBus	FieldBus interfaces	Interface to superior devices
Battery	RAM-backup	Expected life length >10 years
RTC	Real Time Clock	Battery backed Real Time Clock

2.3.1.1 PTC Description

All indicators are located on the front of the PTC:



The following LED indicators can be found on top of the PTC and are common with the LEDs found on regular TCs.

- **OK** is lit green when last cycle (tightening) performed by the PTC was OK.
- **NOK** is lit red when last cycle (tightening) performed by the PTC was NOK.
- **ALARM** is lit red when a servo or spindle hardware or system error has been detected on the PTC. To acknowledge the alarm the PLC variable RESET can be used, an attempt to acknowledge the alarm is also performed automaticly at each cycle start.

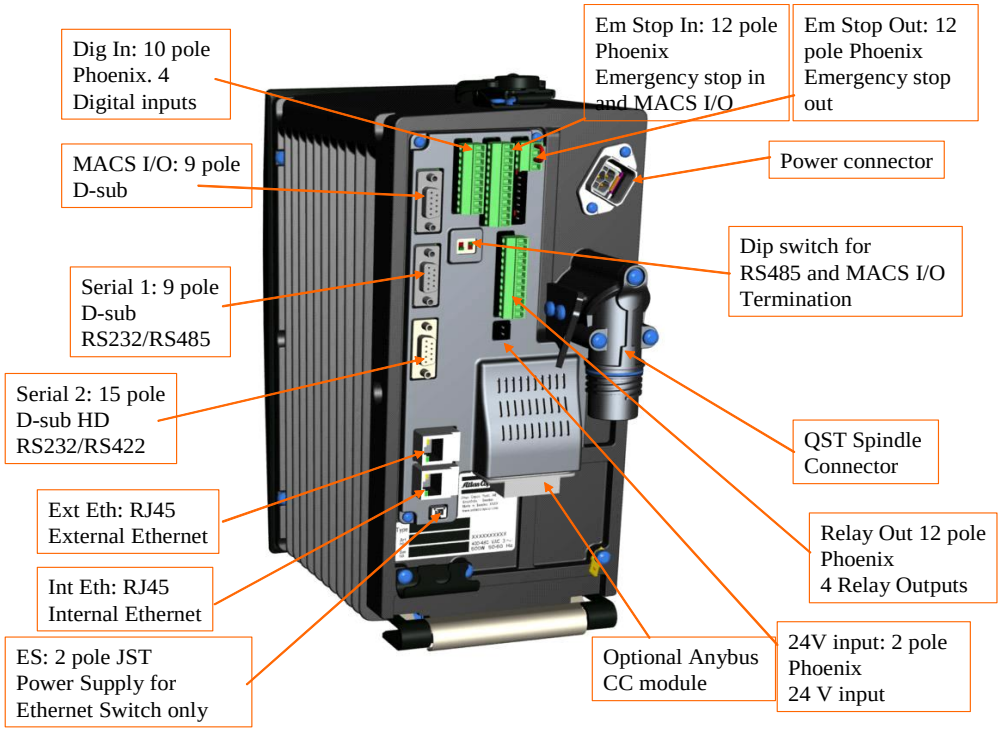
The second row of LED indicators can only be found on the PTC.

- **ALL OK** is PTC specific and indicates that the cycle (tightening) for all bolts for the station was OK.
- **PRIM.** Indicates that the PTC is also the System TC for a system.
- **E-STOP** is lit when the station controlled by the PTC is emergency stopped.

The Reset emergency stop button acknowledges an emergency stop for a station.

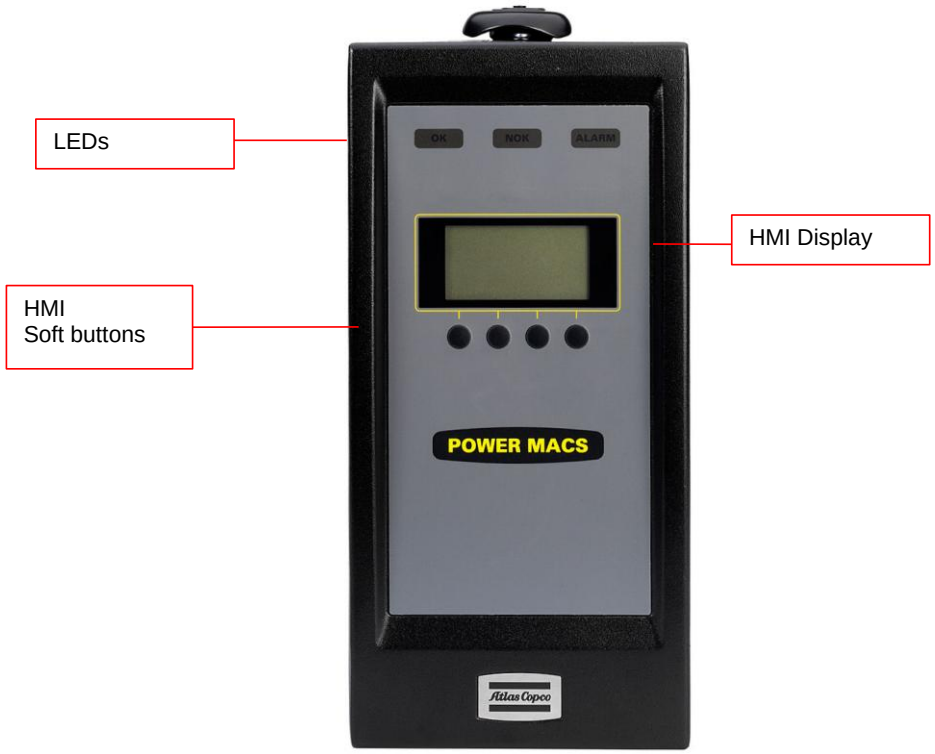
Sockets on the PTC

The following picture shows the input and output sockets on the PTC.



2.3.1.2 TC Description

All indicators are located on the front of the TC.

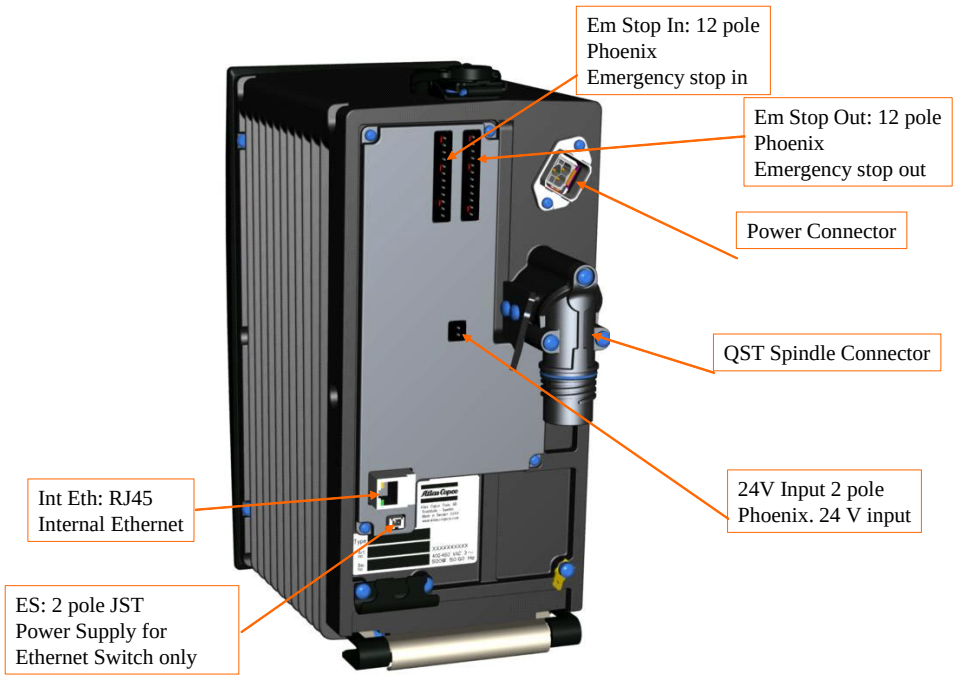


The following LED indicators can be found on top of the TC and are common with the LEDs found on PTCs.

- **OK** is lit green when last cycle (tightening) performed by the TC was OK.
- **NOK** is lit red when last cycle (tightening) performed by the TC was NOK.
- **ALARM** is lit red when a servo or spindle hardware or system error has been detected on the TC. To acknowledge the alarm the PLC variable RESET can be used, an attempt to acknowledge the alarm is also performed automaticly at each cycle start.

Sockets on the TC

The following picture shows the input and output sockets on the TC.

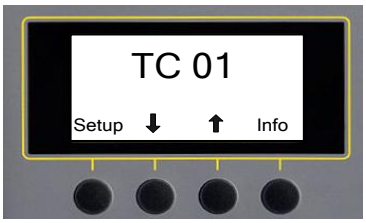


2.3.1.3 The TC HMI Menu System

The TC HMI menu system provides a flexible and intuitive interface to configure and interact with individual TCs. The TC HMI also show the status for certain operations, these include downloading of software, copying or restoring the setup from System TC to a Spindle TC and possible version conflicts detected in the system.

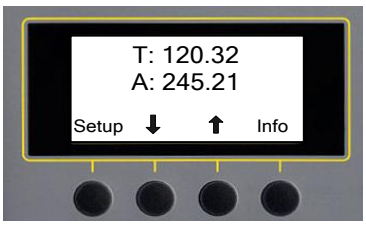
Default Menu

The default menu contains two view items, you can select which view by pressing the **up/down** buttons (the two middle soft buttons). Press the **Setup** button to go to the **Setup** menu and the **Info** button to go to the information menu.



Default view

Shows TC Number (Last digits of internal IP address)



Torque view

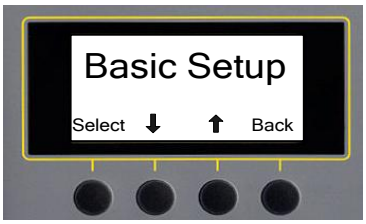
Shows last measured Bolt T and Bolt A

The torque and angle values shown are collected from the monitoring buffer – select the monitoring buffer to use by adjusting the properties on the TC in the System Map, see chapter:

TC node. The selected default menu is remembered between TC restarts.

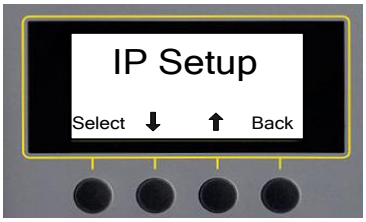
Setup Menu

The setup menu is divided into two group **Basic Setup** and **IP Setup**. Press the **Select** button to enter a settings group or press the **Back** button to return to the **Default menu**.



Basic Setup

Basic settings for a TC



IP Setup

IP Address settings for TC

Setup Menu – Basic Setup

The Basic Setup menu contains a number of settings. To browse between the settings press the **Up**/**Down** buttons. To edit a setting press the **Change** button, when editing it is possible to **Store** the new value or **Cancel** the editing. The picture below shows an example editing the **TC Number** property on a TC.

Edit Settings Example

Select Change to edit

Edit Settings Example

Press Up/Down to Modify
Store or Cancel when done.

The following options can be set in Basic Setup.

Menu	Description
System TC	If a PTC is the System TC (option only available on PTC)
TC Number	The TCs number (equivalent to the TCs last digit in the internal IP-address).
Clear Setup	Clear the setup stored in the TC
Restart TC	Option to restart the TC

Setup Menu – IP Setup

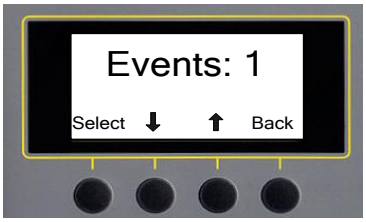
The IP Setup is used to configure the IP addresses of the TC. PTC:s are equipped with two interfaces (internal and external) whereas regular TC:s only have one interface (internal).

The following options can be set in IP Setup.

Menu	Description
IF 1 IP Address	IP Address for interface 1
IF 1 Net Mask	Net Mask for interface 1
IF 1 Gateway	Default gateway for interface 1
IF 2 IP Address	IP Address for interface 2 (PTC only)
IF 2 Net Mask	Net Mask for interface 2 (PTC only)
IF 2 Gateway	Default gateway for interface 2 (PTC only)

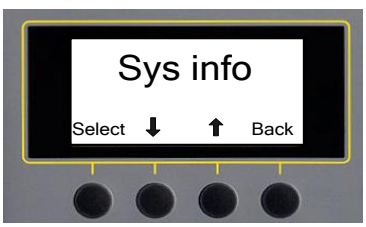
Info Menu

The Info menu contains information about the TC, press the **Select** button to go to either events or system information.



Events Menu

Lists system events



System Information

Shows Information about TC

Info Menu – Events

The events menu contains all hardware and critical system errors that have occured on the TC since the last reset. When the ALARM led is lit on a TC the error message can be found in the events menu.

Show Event

The event code and a short description is shown. The time of the event is shown in the lower left part of the screen.

Press Up/Down buttons to scroll between events.

Info Menu – Sys info

The System Information menu is grouped by TC, Servo and Spindle.

Information Menu	Description
TC Info	
System Type	System Type (PowerMACS Tightening)
Appl Ver	PowerMACS Application Version
Boot Ver	PowerMACS Boot Version
PCB	PCB Version
Hw Ver	Hardware Version
Unit Serial No	Unit Serial Number
Unit Art No	Unit Article Number
TC serial No	Serial number of TC board
IF1 MAC Address	MAC Address of interface 1
IF2 MAC Address	MAC Address of interface 2
Setup Name	Name of currently loaded setup
Servo Info	
Software Ver	Servo Software Version
Boot Ver	Servo Boot Version
Serial No	Servo Serial Number
HW Ver	Servo hardware version
Spindle Info	
Software Ver	Spindle Software Version
Boot Ver	Spindle Boot Version

HW Serial No	Spindle Hardware Serial Number
HW Ver	Spindle Hardware Version
Serial No	Spindle Serial Number
Type	Spindle Type

2.3.2 Version conflict message

All TCs in a system, System TCs as well as Spindle TCs, **must** use the same TC System Software version in order to function properly. There are several methods to verify that this is the case for all TCs in a system. The TCs themselves checks for version conflicts at start up and indicates this using the panel on the their front and TTPM has several forms that can be used for the same purpose (see chapter: Configure Target System and Check for System Conflict in the manual).

The information on the TC panel has been improved to better display to the operator that a Spindle TC does not have the same software version as the System TC. When there is a version conflict with software versions 7.3.0 or higher the Automatic download function takes care of that problem and the messages shown is as described in the chapter Automatic Download.

If a Spindle TC with software version 7.3.0 or higher is inserted in a system with a software version prior to 7.3.0 then the following message is displayed on the TC panel:

Version conflict.

Shows the version of the system TC.

This message is only for information and shows the software version of the System TC, by using the up and down arrows the TC panel could be used as usual.

Note: This message could also appear with Automatic download function as described in the chapter Replacement of System TC and *Backup TC*.

2.3.3 Console Computer (CC)

The *Console Computer* is used for set up of the system and for monitoring and controlling using the ToolsTalk PowerMACS program. It is optional for normal running.

The console computer comprises a standard IBM PC-compatible computer with Microsoft Windows XP operating system (or later) and is not location dependent.

2.3.4 Ethernet

Communication between the *Console Computer* and the *Tightening Controllers* or between two Tightening Controllers is done with use of an Ethernet network. This runs with 100 Mbit/s and uses TCP/IP as protocol. Physical media is 10Base-TX, i.e. twisted pair, max length 100 m. By use of commercially available products it is possible to build a network with optically isolated components.

Via a router other computers in the factory can access the Console Computer. To isolate the PowerMACS system from the rest of the factory (and vice versa) communication is done through an internal separated network, only PTCs provides a second ethernet interface to be used for communication with external equipment.

2.3.5 Peripheral Devices

The following peripheral devices can be connected to the TC or Console Computer:

- I/O device
- Printer on CC (Console Computer running ToolsTalk PowerMACS)
- Printer on TC
- ID device
- Communication via serial protocols
- Communication via fieldbus interface
- Communication from an external PC based application using the PowerMACS API, Application Programmers Interface.

Except for the devices that are to be connected to the Console Computer, most devices can be connected to any PTC tightening controller within the system. For a more complete description of peripheral devices see chapter: Peripheral Devices.

2.4 Automatic update of TC, Servo and Spindle

2.4.1 Automatic update Overview

All TCs in a system, System TCs as well as Spindle TCs, **must** use the same TC System Software version in order to function properly. There are several methods to verify that this is the case for all TCs in a system. The TCs themselves checks for version conflicts at start up and indicate this using the display on their front and ToolsTalk PowerMACS has several forms that can be used for the same purpose.

As from version 7.3.0 it is no longer necessary to use the TTPM to download new software to a replaced TC in a system. When replacing a TC in a system the correct TC software would be downloaded automatically to the replaced TC from the *System TC* or the *Backup TC* (TC2).The software that is downloaded consists of the following:

- The TC application software
- The Servo application software
- The Spindle application software

The automatic download function makes it possible to replace any TC in a system with a TC that has different software loaded, and to automatically update the software in that TC with the same as the rest of the system.

2.4.2 General function

The purpose of the Automatic Download function is to ensure that all TC in a system has the same software version after a TC has been replaced. When a new TC is inserted in a system the version handling function detects if the TC has a different software version. If the software in the replaced TC supports Automatic Download (version 7.3.0 or newer) the download of new software is started when an operator initiate it through the replaced TC:s panel.

The Automatic Download function is always enabled in systems that support it and consist of two or more TCs. Normally all software updates is initiated by the *System TC* that uses its own software to update all other TC:s in the system, if however the *System TC* is replaced it will be updated with the software from the *Backup TC (TC2)* before any other TC is updated.

Before the replaced TC is inserted in the system it must be prepared by an operator to have the right TC number and the right type (P or S). When a replaced TC with a different software version is detected the following text appears on the TC panel:

Fig 1.

Version conflict.

Press Yes if this is replacement TC. Press No if not.

If the operator presses **No**, no software update is made and the TC panel shows the normal version conflict text. (If the operator by mistake has answer **No** to the above questions the TC has to be restarted (power on/off) in order for the questions to appear again.)

If the operator presses **Yes**, the following control text appears on the TC panel:

Fig 2.

Version conflict

Press Yes if this TC should be updated. Press No to return to the previous level.

If the operator presses **No**, no software update is made and the TC panel shows the text in Fig 1.

If the operator presses **Yes**, the software downloads starts and the TC panel shows the normal download information text. If by any reason the download process to any TC is interrupted by loss of power or network error etc. then a TTPM must be used to update these TC since the Automatic Download function would not be a part of the boot loader.

2.4.3 Replacement of the System TC and Backup TC

If the *System TC* detects a conflict compared with the *backup TC* then this situation must first be resolved before updating any other TC in the system. If the *System TC* detects that a *backup TC* with different software version exists in the system then there is no possibility to automatically detect which TC that has been replaced.

The solution to this is to show the above described text on both the *System TC* and the *backup TC* and let an operator decide which TC to update.

Any other TC in the system will show the normal version conflict message as long as there is a conflict between the *System TC* and the *backup TC*.

Note: It is not possible to download new software to the *backup TC* from a *System TC* that has no valid setup.

2.4.4 Replacement of TC3 – TCn

If the *System TC* and *backup TC* contains the same software version and has a valid setup the *System TC* continues to check for any software version conflict among TC3 – TCN. If one or several Spindle TCs are found with a different software version the above described dialog appears on all the replaced TC. The Spindle TC then starts to download the new software from the *System TC* by using functions provided by the *System TC*.

Note: Since the *System TC* only initiates the download of new software any number of Spindle TCs could download new software simultaneous without any interference from the *System TC*.

2.4.5 Updating a system with new software

As described in the previous chapter it would be possible to update a whole system with new software by only download it to the *System TC* and then let the Automatic Download function update the rest of the TCs in the system.

Note: For this to be possible the *System TC* must have a valid setup loaded.

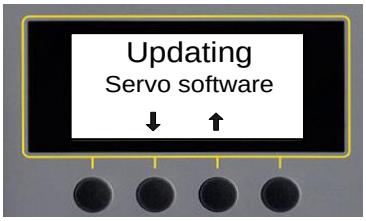
2.4.6 Troubleshooting

If by any reason the download software process to any TC is interrupted by loss of power or network error etc. then a TTPM must be used to update these TC since the Automatic Download function would not be a part of the boot loader.

2.4.7 Automatic Update of Servo software

When the TC is powered up a control is made to verify if the software that is running in the servo is different than the servo software stored in the TC flash memory. If it is found to be different and older that the software stored in the TC the servo is automatically updated with the newer version.

During the update of the servo software the following message is shown on the TC panel:



Updating

Indicates that the TC is updating the servo software..

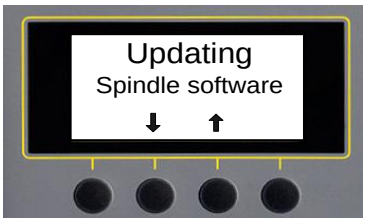
It is also possible to downgrade a servo with software that is older than the version running in the TC. This could be done by using the Configure Target function described in this release note.

Note: During the start and end of the servo updating the servo switches the power to the TC off and on.

2.4.8 Automatic Update of Spindle software

When the TC is powered up or when a spindle is connected a control is made to verify if the software that is running in the spindle is different than the spindle software stored in the TC flash memory. If it is found to be different and older that the software stored in the TC the spindle is automatically updated with the newer version.

During the update of the spindle software the following message is shown on the TC panel:



Updating

Indicates that the TC is updating the spindle software..

It is also possible to downgrade a spindle with software that is older than the version running in the TC. This could be done by using the Configure Target function described in this release note.

Note: If the updating of the spindle software should fail due to that the power is switched off or the spindle is disconnected before the download has been completed, then the spindle needs to be updated manually by using the Configure Target function.

3 Basic Functions

3.1 Basic Functions - Overview

This part covers the basic functions, like how to handle windows and screen, security, help, etc.

3.2 Windows

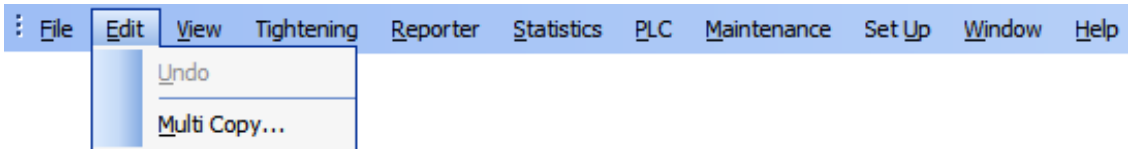
This chapter gives a general presentation of the parts of ToolsTalk PowerMACS and how to operate them.

Title bar

On top of the screen there is a *title bar*. In this you can see the name and version of the application (ToolsTalk PowerMACS 7.3.0) and the name and version of the setup that you currently are working with. If no specific setup is active <no name> is displayed instead of the system name.

Menu bar

On top of the screen, just below the title bar, there is a *menu bar*.



When selecting one of the menus it drops down and you can see a set of *menu items* or *menu choices*.

Tool bar

Below the menu bar you have a *tool bar*. This contains buttons to reach the most frequently used functions of the system. You can move the toolbar to a different location by dragging the toolbar using the grip-handle on the left side.

Status bar

On the bottom of the screen you have a status bar.

This is used by the system to show (from left to right):

- Status fields (information from the system online, offline, errors, problems etc.)
- Target (IP address of System TC)
- Current user logged in (”Anonymous” if no one has logged in)
- Current date and time

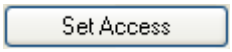
Commonly used Windows controls

The following Windows controls are commonly referred to in this document:

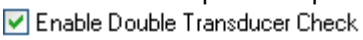
- **Textbox** – Holds text that you can either enter or change

- **Frame** – Groups a number of controls that belongs together

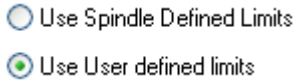
- **Button** – Carries out a command when pressed



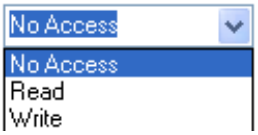
- **Check box** – Represents parameters of Boolean type, that is that either are True or False



- **Radio button** – Represents multiple choices from which you can choose only one



- **Combo box** – Another way to represent multiple choices from which you can choose only one



- **Spread** – An Excel style control used to display table oriented data

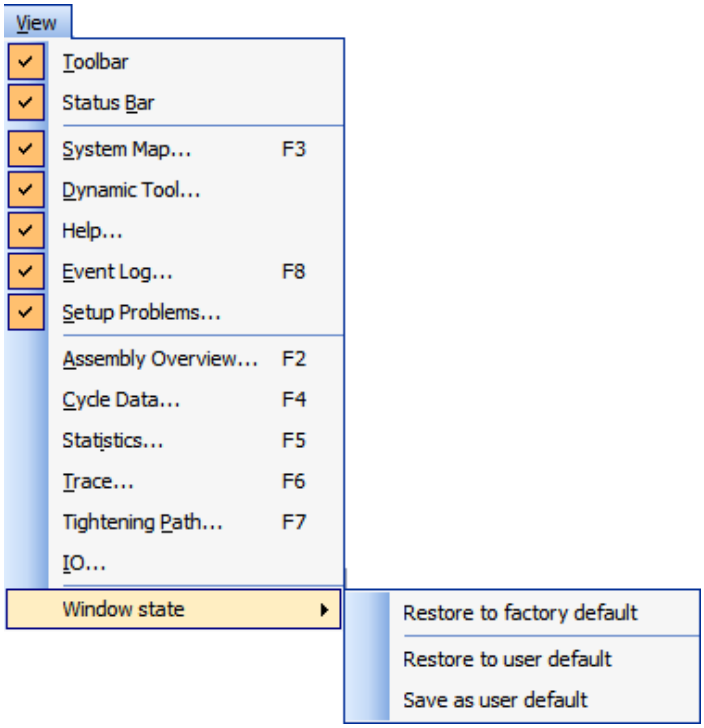
Mode	1	2	3
Name	Mode 01	Mode 02	Mode 03
Bolt 01	Loosening	Loosening	Fast

- **Property grid** – A list of properties for an object

Step type	A - Run until Angle
Syncronize	☑
Start/restart mon	☐
Speed (rpm)	30
Direction	Backward
Angle (deg)	Angle
Start Condition	Step Start (AS1)

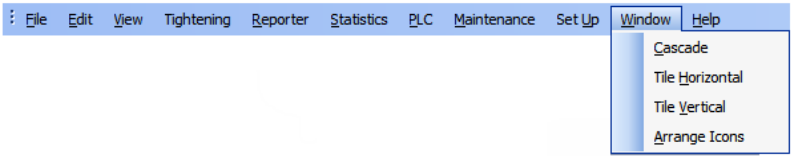
Handling of windows

With the menu View you can select which dockable windows you want to show and also if the toolbar and/or the status bar should be displayed.



It is also possible to restore the factory default view from here and save your own view configuration..

If you have a number of windows on the screen at the same time you can get these rearranged in a structured way. Use the menu choices **Window-Cascade** etc. Note that these options will not scale the dockable windows.



The layout of most windows are saved when ToolsTalk PowerMACS is closed and will be restored when ToolsTalk PowerMACS is started again.

3.3 Help

Function key F1

When running the ToolsTalk PowerMACS application you can get all information you need, by a press of a button.

Whenever in trouble, press the function key F1

F1 activates the help system and displays information relevant the topic you are just using. This is called context sensitive help.

Automatic Help Window

The automatic help window on the right side of ToolsTalk PowerMACS will automaticly display detailed help about certain items you are working on. For example it will automaticly show the help text for the selected step in the tightening window.

Help menu

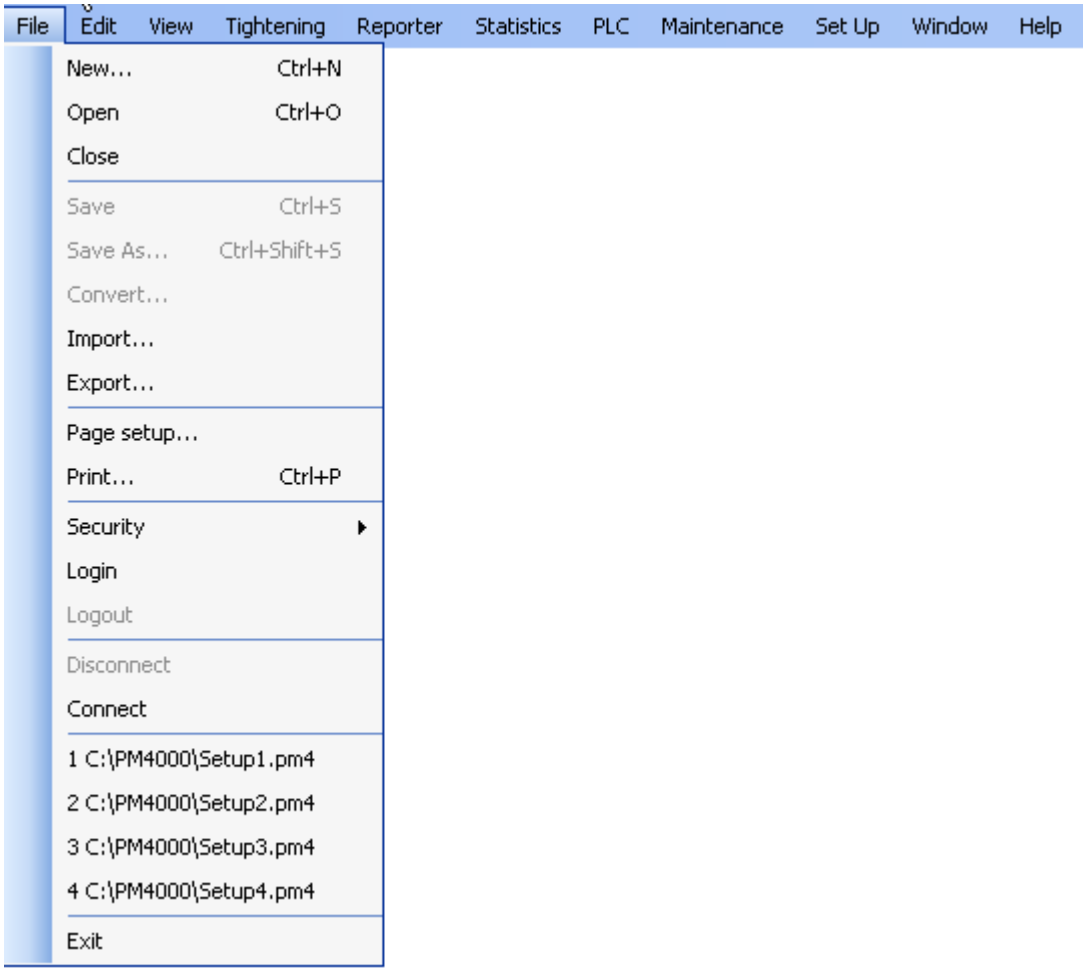
It is also possible to get help by use of the **Help** menu. **Contents** brings up a list of topics with hyper links. **Search for help on…** brings up a dialog box where you can enter a keyword to the topic you want help on.

Troubleshooting will present a guide that can assist you in case of trouble. **Maintenance** contains information on preventive maintenance, spare part lists etc.

Site Specific Help presents information that is specific for the current site.

3.4 The File menu

On the **File** menu you can find basic functions for ToolsTalk PowerMACS.



The items **New**, **Open**, **Close**, **Save**, and **Save as** are for handling setups. **Convert** is used to convert an old setup to a newer version. **Import** and **Export** are also used in connection with setups. Handling setups are described in chapter: Set Up and Maintenance.

Page setup and **Print** is used for printing of setups. See chapter: Printing.

Security, **Login**, and **Logout** is described in detail in chapter: Security.

Connect, Disconnect indicates if ToolsTalk PowerMACS is in contact with a running system. If you want to work off-line, even though you have a connection, select this item to switch it off. See also chapter: How to Start up.

An **MRU** list (Most Recently Used) shows previous setups that has been opened recently. Select one of these to get an instant opening of it, instead of using the **Open** item.

Use Exit to shut down the ToolsTalk PowerMACS application. If you have modified the current setup you will be asked to save it first.

3.5 How to Start up

Depending on what you want to do you start ToolsTalk PowerMACS in either Connected or Disconnected mode.

If ToolsTalk PowerMACS was connected when it was last shut down it will try and reconnect to the same system upon startup.

The Welcome screen

When ToolsTalk PowerMACS is first started it will present a **welcome screen** containing the most common starting points for working with ToolsTalk PowerMACS.

From this window it is possible to select **Create a new setup** to begin working on a new setup in disconnected mode, select **Open an existing setup** to open an already saved setup from disk. The option **Select a target system** allows you to select the target system to connect to, to connect to the target system most recently used select **Connect to current target system**. You can also select one of the most recently used setups to open in disconnected mode.

If you do not want the **welcome screen** to be shown when ToolsTalk PowerMACS is started check the **Do not show this again** checkbox before closing the window. The **welcome screen** can be enabled again in the **Setup -- Options..** menu.

To edit a setup when Disconnected

This is how you start up when you want to check or edit a setup, without connecting to a running system:

1. From the **File** menu select Open, or from the **welcome screen**, select **Open an existing setup**
2. In presented file list, select an existing setup

If you want to create a completely new setup, select **New** from the **File** menu or **Create a new setup** from the **welcome screen**.

To connect to the current target system

This is how you start up ToolsTalk PowerMACS when you want to connect to the current target system (that is the system ToolsTalk PowerMACS was most recently connected to).

Using the **welcome screen**:

1. Make sure your console computer is connected to the network of the system in question.
2. Press the **Connect to the current target system** to connect.

Using the regular interface:

1. Make sure your console computer is connected to the network of the system in question.
2. Press the **Connect** button on the toolbar.

To connect to a specified system

This is how you start up ToolsTalk PowerMACS when you wan to connect to a specified target system.

Using the **welcome screen**:

1. Make sure your console computer is connected to the network of the system in question.
2. Press the **Connect to system..**
3. From the **Select Target System** window select the system.

Using the regular interface:

1. Make sure your console computer is connected to the network of the system in question.
2. Press the down-arrow on the **Connect** toolbar button and select **Select target system...**
3. From the **Select Target System** window select the system.

If you want to download a setup to the target fist make sure it is loaded in ToolsTalk PowerMACS, either by reading it from disk (menu: **File – Open**) or by Creating a new system using the Set Up Wizard (menu: **File – New**).

Connecting to a system

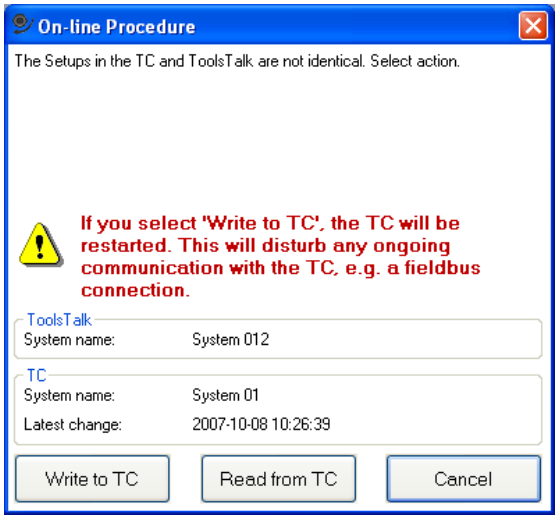
When ToolsTalk PowerMACS connects to a system over the network the following can occur:

If the setup version and the TC software version differ, you will be notified and if there is a solution to the version conflict a solution will be suggested.

If there is no setup loaded in ToolsTalk PowerMACS the setup will automaticly be fetched from the System TC and ToolsTalk will connect normally.

If a setup is present in ToolsTalk PowerMACS and it is equal compared to the setup in the System TC ToolsTalk will connect normally.

If a setup is loaded in ToolsTalk PowerMACS and it is *different* from the setup stored on the TC the following dialog is displayed:



Select **Write to TC** to transfer the setup currently loaded in ToolsTalk PowerMACS to the target system. The target will wait for any ongoing cycles to be completed before the setup is loaded. Should the cycle not finish within time the download is canceled and you are informed.

Note! Selecting **Write to TC** will restart the target system. This may disturb any eventually ongoing communication that the target has with other systems, for example over a fieldbus.

Select **Read from TC** to transfer the setup in the target system to ToolsTalk PowerMACS. Please note that this will replace the setup currently stored in ToolsTalk PowerMACS.

3.6 Viewing

If you have your console PC connected to the tightening controllers (connected), you can view several items in the system. You can start View displays by use of the menu **View** or by pressing one of the corresponding buttons on the tool bar.

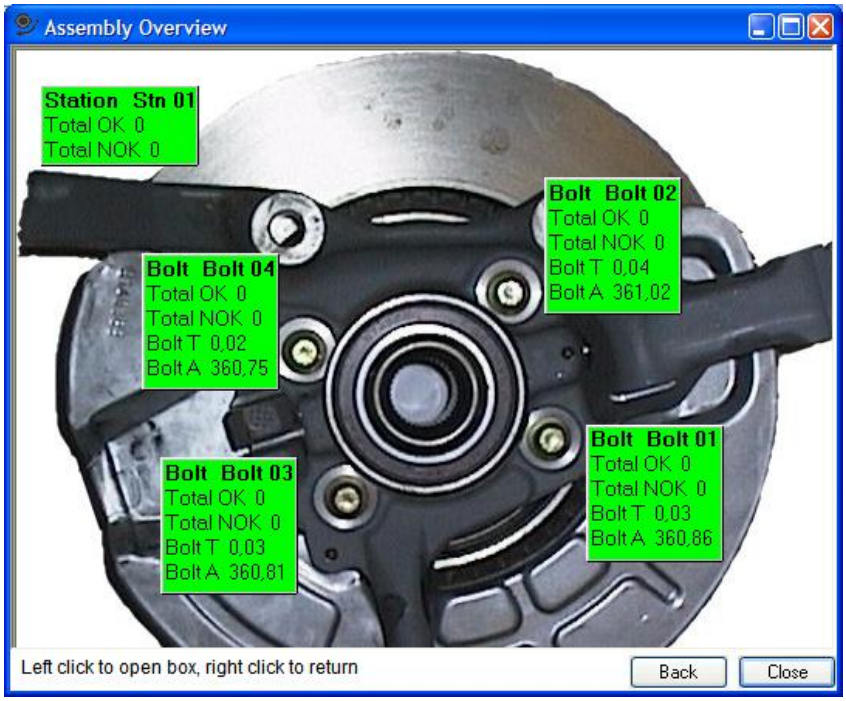
3.6.1 Assembly Overview

The Assembly Overview displays current status of the system, its stations and bolts using a descriptive picture of the object being assembled. You can easily add pictures as background for the system and the respective station. See chapter: Assembly Overview Set Up form for instructions.

The color of the items indicates the current status of the item.

Color	Status
Green	OK - Idle after OK cycle
Red	NOK – Idle after NOK cycle
Orange	NOKRM - Idle NOK only due to reject management after a cycle. For bolts only
Yellow	Running cycle
Blue	Bolt is disconnected. This color is only used if the checkbox **Display disconnected bolts with blue color** in the Options form is checked. If not checked then disconnected OK bolts will be green and disconnected NOK bolts will be red. For bolts only

If you want more information on the stations or bolts, expand the box by clicking on the box of interest.



The information that is displayed in the different boxes and the size and position of them is set up using the Assembly Overview Set Up form.

If you have several stations in your PowerMACS system they can all be displayed in expanded mode over a common system background picture.

When a cycle is started, the station box and bolt boxes (for running bolts) are cleared of its result information and the background is set to yellow. For bolt boxes of bolts that are not started the previous

result information and background color remains. When the cycle is finished, the background color and result information is updated.

In the case that the cycle is ended without any cycle data being produced (for example a sub cycle is ended in a stitching cycle using PLC station output variable DATAHOLD); the **bolt** boxes background color is updated to show the **status** of the finished cycle.

Note! The station box is still yellow to indicate that the complete tightening cycle is not yet finished and that there is no result information in the bolt boxes. The reason for this is that the cycle data is not yet produced by the system and therefore it is not possible to show any result information either. When the complete cycle is finished and the cycle data is received, the station and bolt boxes are updated with status and result information.

3.6.1.1 Station Views

For a station, different background pictures and layouts of the station and bolt boxes can be used. The currently selected Mode number triggers which so called Station View to display. The mapping between Station Views and Mode numbers are done using the Assembly Overview Set Up form.

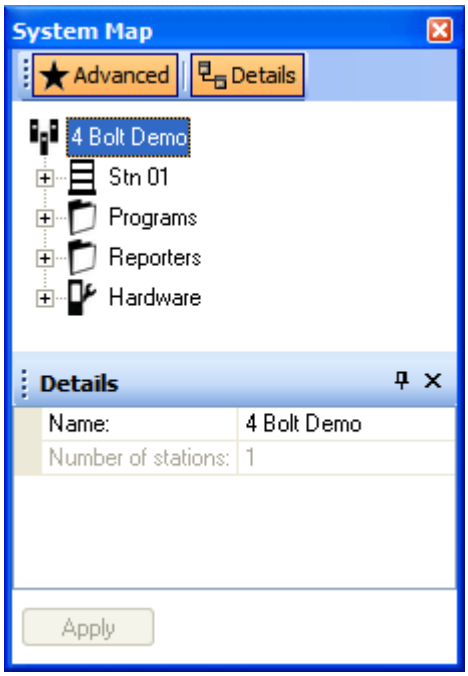
The Station View change is triggered by a change in the Mode number. That means that if the Mode number is changed before a cycle is started, the Station View is also changed before the cycle is started. Changing Mode number is done by using the PLC station output variable MODE (see Station variables). If a cycle is running when the Mode number changes, the Station View is not changed until the cycle is finished.

If the Mode number changes to a Mode number that does **not** map to any Station view (or the Mode number is reset to zero), the currently shown Station View remains.

When the Station View changes, the status color and result variables of all station and bolt boxes remains. If it is desired to clear these values, use the PLC station output variable RESETSTATUS (see Station variables). A positive edge on this variable will cause all result information in Assembly Overview to be cleared and set the background color of all station and bolt boxes to grey. The status information in the System Map will also be cleared.

3.6.2 System Map

The System Map gives an overview of all functional parts in the system. It can be used both as an indicator of current status for all parts, but also as a navigation tool within the system. The default location for the System Map is on the left side of ToolsTalk PowerMACS but it can be placed anywhere on the screen.

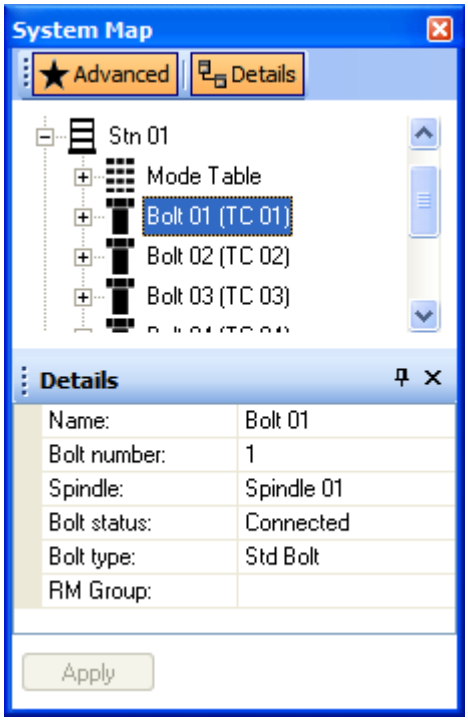


When first started the System Map displays a page with the system shown in a tree structure. You can expand and collapse the tree structure by pressing the small squares with **+** or **-**, or clicking on items.

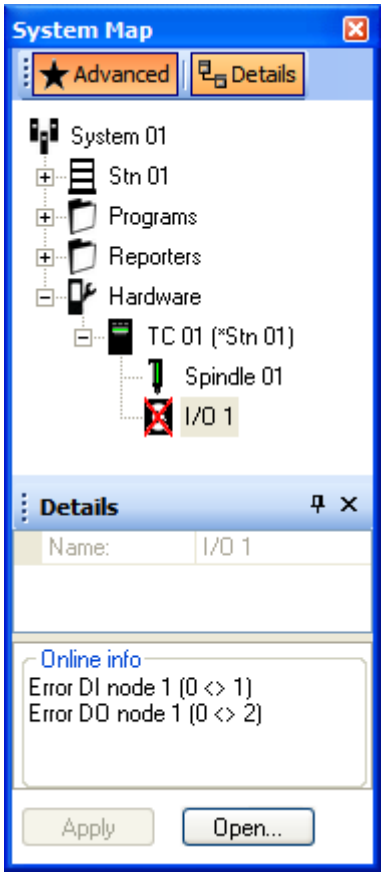
When connected to a system the Station and Bolt nodes will display certain colors depending on the last status.

Color	Status
Green	OK - Idle after OK cycle
Red	NOK - Idle after NOK cycle
Orange	NOKRM - Idle NOK only due to reject management after a cycle. For bolts only
Yellow	Running cycle
Blue	Bolt is disconnected. This color is only used if the checkbox **Display disconnected bolts with blue color** in the Options form is checked. If not checked then disconnected OK bolts will be green and disconnected NOK bolts will be red. For bolts only

Below the actual tree are the particular **Details** for the currently selected item listed. Use the **Details** window to edit data for an item. By default the Details window is placed below the actual system tree but it can be placed anywhere on the screen by drag & drop.

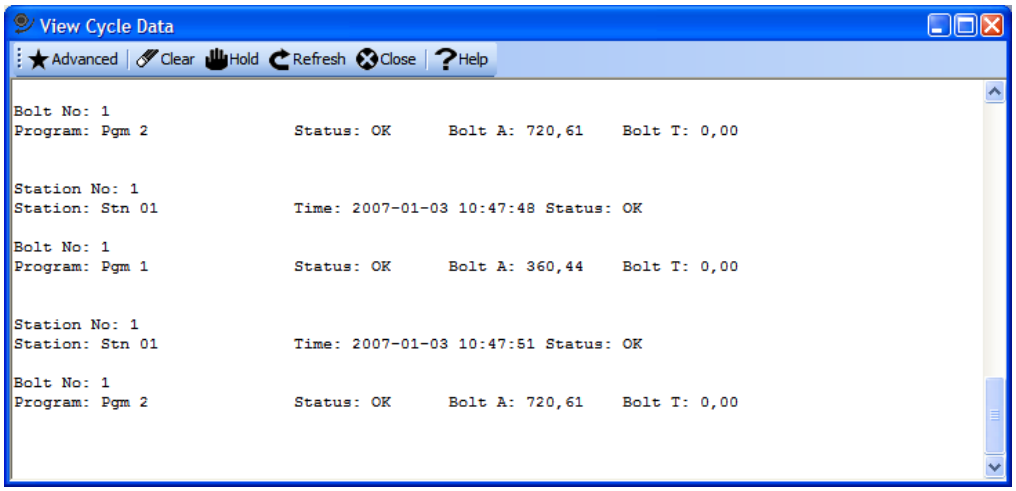


Problems with equipment (hardware or devices) are indicated with a red cross over the icon. For certain devices Online info is shown below the Details window.



3.6.3 View Cycle Data

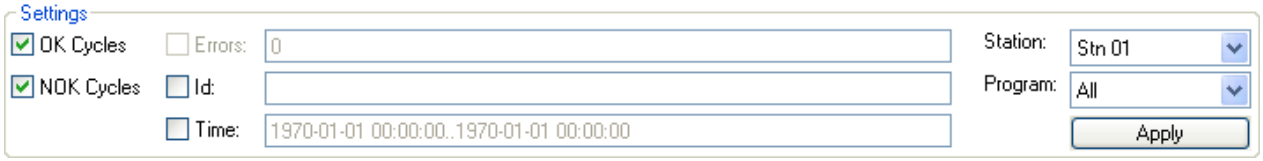
The Cycle Data window is used to display data from the tightening cycles.



The display shows the last cycle. When new cycles are added older cycles are scrolled upwards. By use of the vertical scrollbar you can go back to earlier cycles. If the text is wider than the window, a horisontal scrollbar appears.

If you press **Hold** the display is temporarily stopped. Cycle data are still recorded and when you release the Hold by pressing the **Hold** button again the presentation will continue. Press **Clear** to clear the list. Press **Refresh** to clear the list and retrieve all cycle data stored on the System TC.

If the **Advanced** button is pressed you can specify the format settings by pressing the **Layout** button, if you want to present just specific cycles press the **Settings** button, the following frame will then appear below the cycle data window:



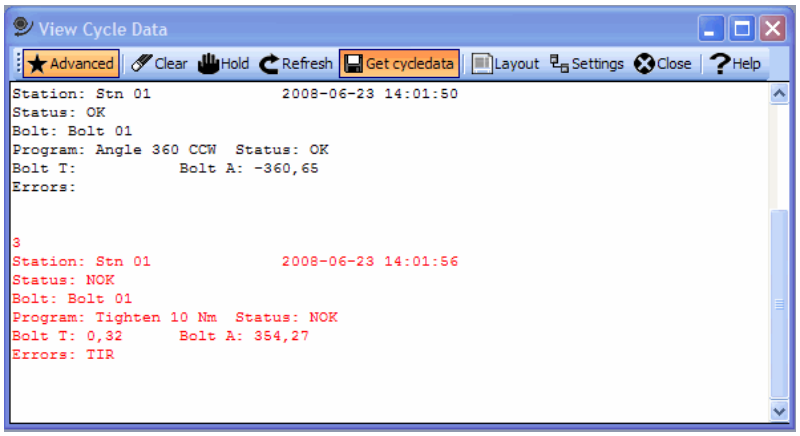
Here you can specify if you want to present OK and/or NOK cycles, for specific stations and/or tightening programs, or cycles with specific errors.

In the ID field write as many characters you like. All cycles that have an ID that contains the string of characters will be displayed.

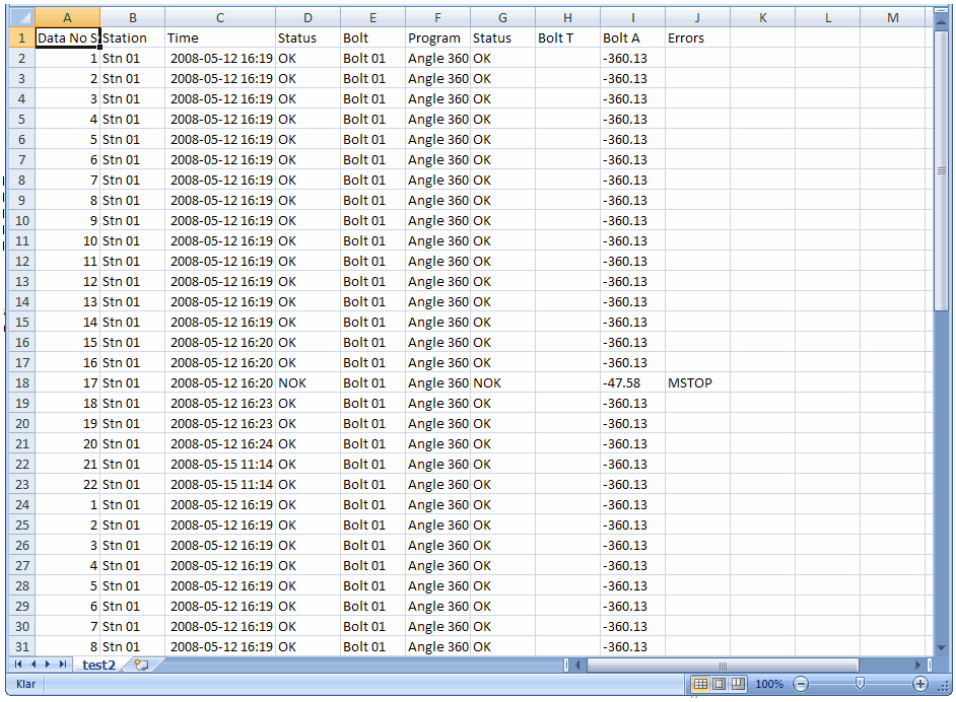
You can also present cycles between certain times, e.g. 99-12-31, 23:59..00-01-01, 00:01. Enter date and time in the format your computer is set up for. Use ".." as separator between start and stop times. If you do not enter any date, today is assumed. If you do not enter any time, start time is assumed to be 00:00 and stop time 23:59.

Note! To be able to filter out the results for a specific program you must have included the bolt level result variable “Program” in the reporter named “Screen”. The result will include all cycles where at least one bolt has executed the selected program.

Cycle data can be saved to file when pressing the “Get cycledata” button. Select a file to save the data to or type a new file name. The data will be saved into this file with the extension .csv



The picture below shows the result when the file is opened with Excel. Please note that the file can have different columns depending of the settings in the screen reporter.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Data No S	Station	Time	Status	Bolt	Program	Status	Bolt T	Bolt A	Errors			
2	1	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
3	2	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
4	3	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
5	4	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
6	5	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
7	6	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
8	7	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
9	8	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
10	9	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
11	10	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
12	11	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
13	12	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
14	13	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
15	14	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
16	15	Stn 01	2008-05-12 16:20	OK	Bolt 01	Angle 360	OK		-360.13				
17	16	Stn 01	2008-05-12 16:20	OK	Bolt 01	Angle 360	OK		-360.13				
18	17	Stn 01	2008-05-12 16:20	NOK	Bolt 01	Angle 360	NOK		-47.58	MSTOP			
19	18	Stn 01	2008-05-12 16:23	OK	Bolt 01	Angle 360	OK		-360.13				
20	19	Stn 01	2008-05-12 16:23	OK	Bolt 01	Angle 360	OK		-360.13				
21	20	Stn 01	2008-05-12 16:24	OK	Bolt 01	Angle 360	OK		-360.13				
22	21	Stn 01	2008-05-15 11:14	OK	Bolt 01	Angle 360	OK		-360.13				
23	22	Stn 01	2008-05-15 11:14	OK	Bolt 01	Angle 360	OK		-360.13				
24	1	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
25	2	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
26	3	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
27	4	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
28	5	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
29	6	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
30	7	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				
31	8	Stn 01	2008-05-12 16:19	OK	Bolt 01	Angle 360	OK		-360.13				

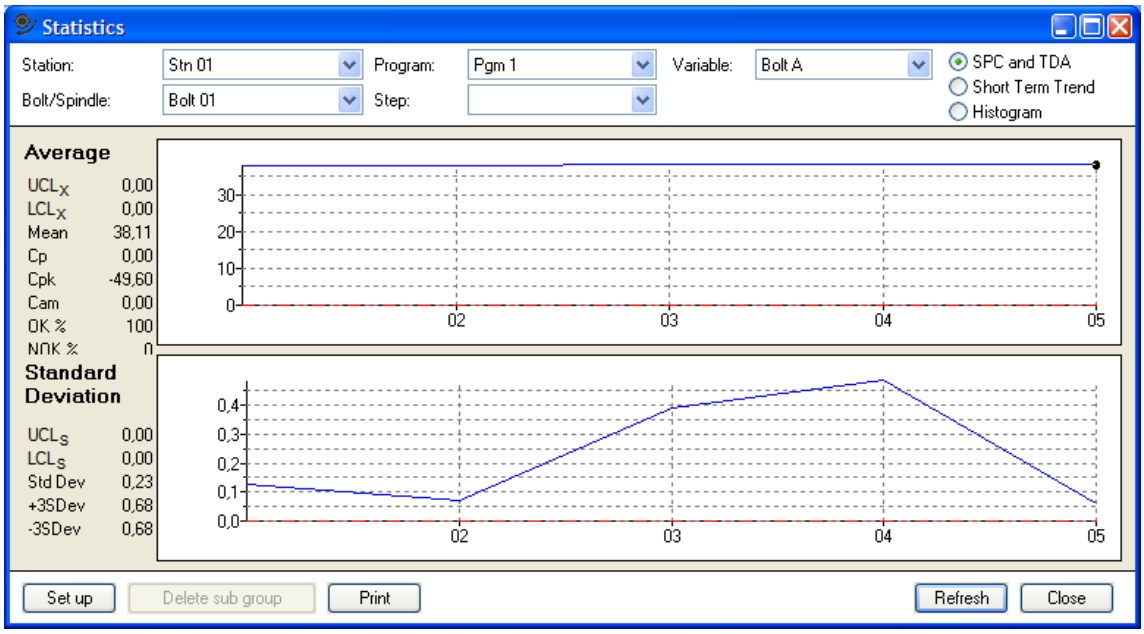
3.6.4 View Statistics

The Statistics window displays the statistical curves generated by the function "SPC, Statistical Process Control".

First select with **Station**, **Bolt or Spindle**, **Program**, **Variable**, and **Step** what variable you want to study the statistics for. Only values earlier configured for SPC using the SPC set up form can be displayed.

Then use the radio buttons in the upper right corner of the form to select which type of display you want. The following types are available:

- Average and Standard Deviation/Range curves for SPC and TDA
- Average and Standard Deviation/Range curves for Short Term Trend
- Histogram



With the cursor you can mark a specific part of the trace to zoom in. Place the mouse pointer on the upper left corner of the area you want to enlarge. Press the left mouse button and drag the mouse pointer to the lower right corner of the area to zoom in and release the mouse button. This will cause the graph to be redrawn displaying only the selected area. Click on the right mouse button to zoom out.

When the mouse pointer is located over the graph area a box under it will display the value of the x and y coordinates pointed at.

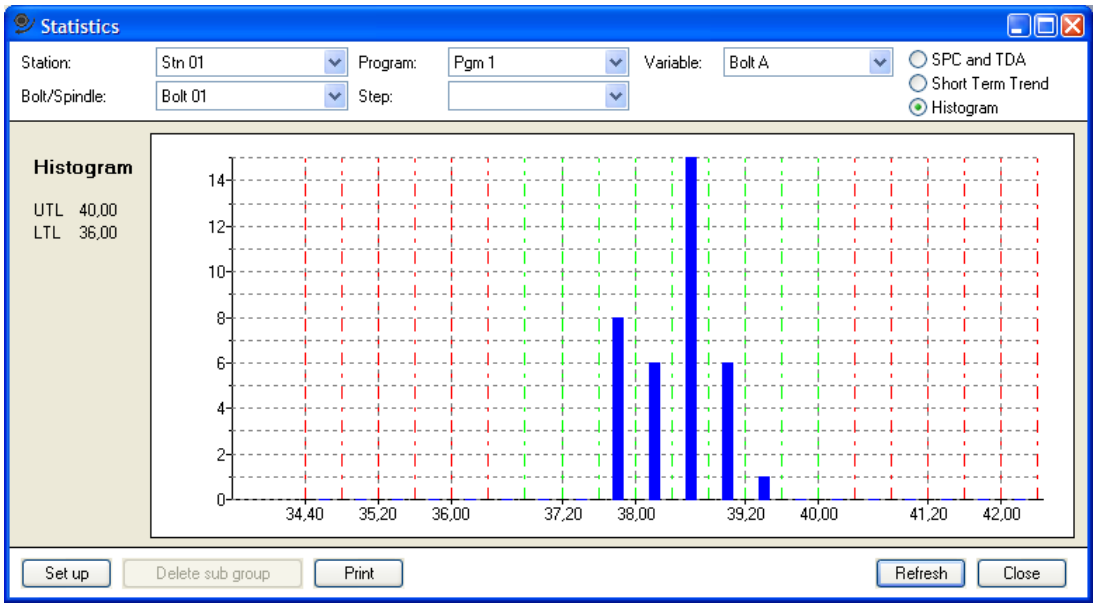
If you select **All Bolts** or **All spindles** the SPC diagram will show you SPC statistics based on data from all bolts or spindles in a station weighed together to a measure of the station as a whole. Only variables already set up for collection from this station are used when calculating the resulting values.

If there is a subgroup, which has possible corrupt data, you can delete it. Click in the diagram on the subgroup you want to delete. Press **Delete Subgroup**. This is possible only when the selection SPC and TDA is active.

To the left current capability values are displayed. The window is automatically updated whenever a new cycle is completed. Values displayed are calculated over same data as shown in the curve, for subgroups or for short-term trend.

If Cp or Cpk is not presently within limits, a warning message is presented.

If you select **Histogram** you will be presented the following view:



This form displays the frequency distribution of the selected variable. The X-axis, displayed in the unit of the variable, is divided in 20 classes scaled so that 25% of the screen is left of LTL, and 25% is right of UTL.

The boundaries of each class are marked using dashed vertical lines where green indicates classes in between LTL and UTL.

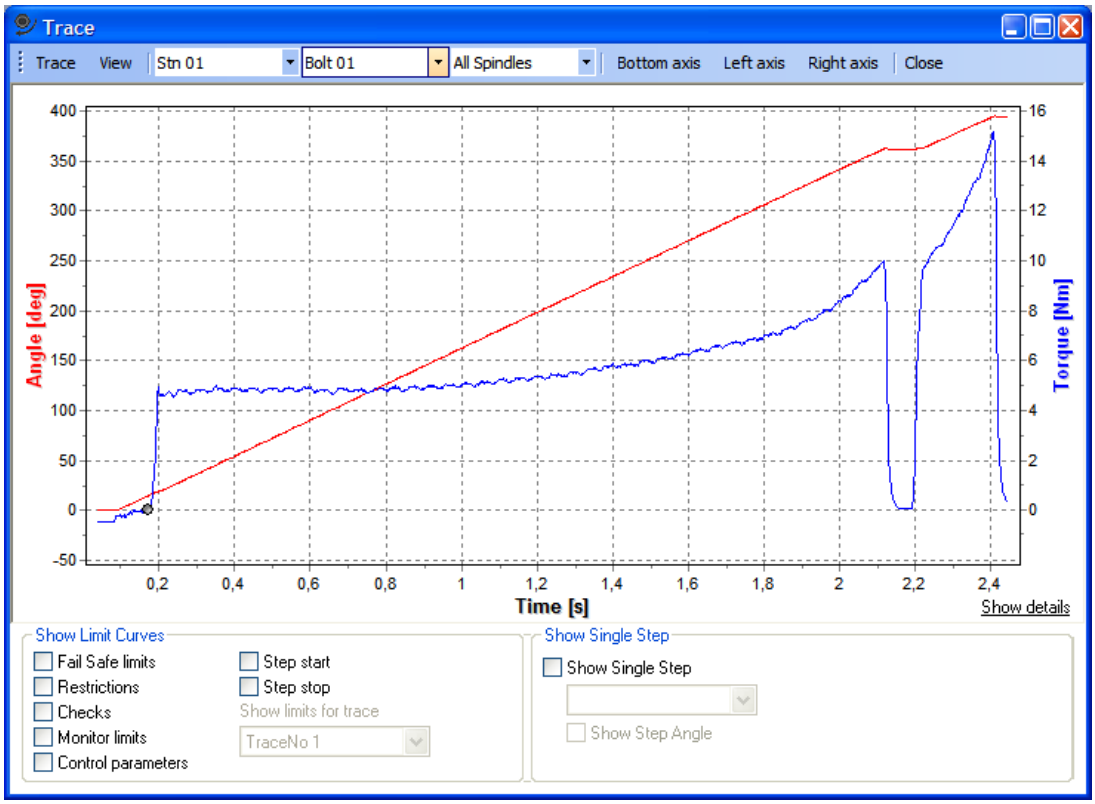
On Y-axis is shown the count of values with in each of the classes.

If you want to have the current view on paper, press **Print**.

The viewed statistics is updated cyclically when online.

3.6.5 View Trace

The Trace viewing window is used to present the traces of current and prior cycles.



For each individual bolt's tightening cycle a trace can be generated and recorded. The maximum length of a trace is 20 seconds. If a trace is recorded for longer time than 20 seconds the oldest values will be overwritten with new samples.

The conditions for when a trace is started is configured under the general settings for each program.

3.6.5.1 Select which trace to display

The controls in the top of the Trace viewing window is used to select which cycles to show traces for and which quantities to show for these cycles. You can also save and open traces from both file and the TC.

Select cycle

Select the bolt you are interested in by use of the **Station** and **Bolt** fields. Select the spindle used to tighten the bolt by use of the **Spindle** field. “All” means that the trace for selected Bolt is viewed regardless of which spindle was used to tighten the bolt.

To view traces for a particular spindle, regardless of which bolt the spindle tightened, select “All” in the **Bolt** field and then select the spindle in the **Spindle** field.

Quantities

Select which quantities you want to display for the trace by use of the three menues on the upper right side of the trace viewing window (Bottom axis, Left axis and Right axis). You can study one or two different quantities of the trace against a third quantity.

For the two vertical axis, the following quantities can be selected:

- **Angle** - Views the recorded angle values.
- **Current** - Views the recorded current values.
- **Current as T** - Views the recorded current values scaled as torque using the values of the spindle setup parameters Gear Ratio and T/C factor from when the cycle was executed.
- **Torque** - Views the recorded torque values.
- **Gradient** - Views the recorded gradient values.
- **Pos. Angle** - Views the recorded angle values filtered so that all angle movements are positive, i.e. the angle of the cycle is always increasing, even if the bolt was moving backwards.
- **Pos. Torque** - Views the absolute values of the recorded torque values.

If Advanced view is enabled it is also possible to select the following for the second angle and torque channels:

- **Angle 2nd** - Views the recorded angle values for the second angle channel.
- **Torque 2nd** - Views the recorded torque values for the second torque channel.
- **Pos. Angle 2nd** - Views the recorded angle values for the second angle channel filtered so that all angle movements are positive, i.e. the angle of the cycle is always increasing, even if the bolt was moving backwards.
- **Pos. Torque 2nd** - Views the absolute values of the recorded torque values for the second torque channel.

For the horisontal axis, the following quantities can be selected:

- **Time** - Displays values over sample time.

- **Angle** - Displays values over recorded angle.
- **Pos. Angle** - Displays values over positive angle. All angle movements are positive, i.e. the angle of the cycle is always increasing, even if the bolt was moving backwards.

If Advanced view is enabled it is also possible to select the following for the second angle channel:

- **Angle 2nd** - Displays values over recorded second angle channel.
- **Pos. Angle 2nd** - Displays values over positive second angle channel. All angle movements are positive, i.e. the angle of the cycle is always increasing, even if the bolt was moving backwards.

For all quantites except angle and time, there may be gaps in the recorded values. For example when spindle functional test is performed, torque values are not recorded. When the torque curve is displayed on the graph, a small gap in the curve is shown when the spindle functional test was performed. Another example is Gradient values which are only recorded when the restriction Gradient is active.

If Torque (or Pos. Torque) is selected, the torque unit displayed is the one used when the trace was recorded. If many traces are displayed with different torque units, the torque unit for each trace can be found in the identity list. This list is displayed by clicking on the button **Identity>>**.

The second torque and angle channels are only available when the following conditions are true:

- Equipped on spindle and marked as enabled in the spindle setup.
- Selected as included in the general program settings.

Note! The actual second Angle and Torque channel used is the channel that is not configured as Monitoring channel (regular Angle and Torque channels).

Note! Collection of Secondary Torque and Angle channels in the trace must be enabled for each program, this is described in chapter Program Settings.

Automatically updated traces

Under the **View** menu it is possible to select if the **Latest** trace should be automaticly displayed or only the **Latest NOK** or **Latest NOK**. When a specific trace is selected the choice is set to **Selected**.

Open and save traces

Using the menu **Trace** is is possible to **select** a trace stored on the TCs or as a disk file on your PC.

- To read a trace from a TC select the **Select from TC** option from the **Trace** menu. This will invoke the Select Trace form with which you can select not only one but several traces, and from one or multiple bolts and spindles.
- To read a trace earlier saved on disk select the **Open file** option from the **Trace** menu. This will invoke a file selection dialogue in which you can select the file to display.

If you see a trace that is interesting in some respect you can choose to save it, either in the TC or to a disk file on your PC.

- To save it in the TC select the **Save at TC** from the **Trace** menu. This will cause the currently displayed trace to be tagged for save which protects it.
- To save it as a disk file select the **Save to file** option from the **Trace** menu**.** This will display a **Save As** dialogue where you specify the name and the format of your file. Chose extension *.trc ("Trace Files"), to save it in PowerMACS internal format, or extension *.txt ("Text Files"), to save the trace as a text file. A trace saved in PowerMACS internal format includes all data of the trace and can later be opened using the **Trace|Open file** alternative. A trace file stored as text only includes the curves, the step boundaries (as a curve) and the cycle data printed as text. The curves are written as columns, separated by a tab character, with each sample on an own row. This makes the text file easy to import to, for example, Excel.

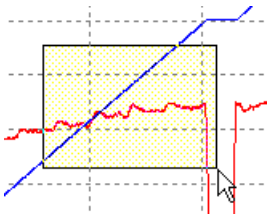
3.6.5.2 View of trace curves

In the middle of the Trace viewing window, the curves are plotted for the selected traces and selected quantities.

Zoom

With the cursor you can mark a specific part of the trace to zoom in. Place the mouse pointer on the corner of the area you want to enlarge. Press the **left** mouse button and drag the mouse pointer to the opposite corner of the area to zoom in and release the mouse button. This will cause the graph to be redrawn displaying only the selected area.

Click on the **right** mouse button to zoom **out**.

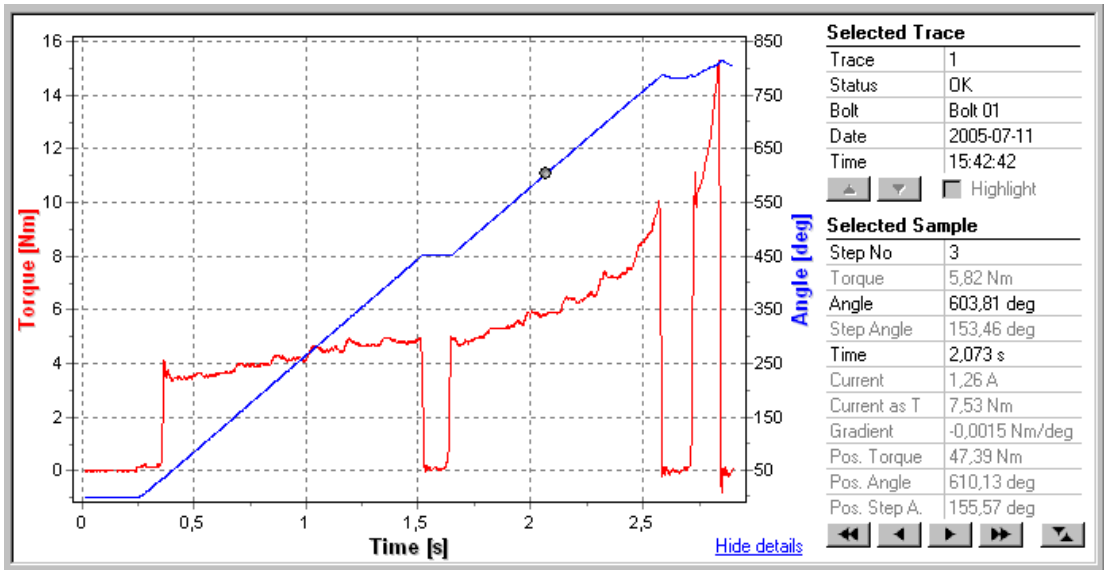


Scroll

When the graph is zoomed in, you can scroll the graph to be able to follow a curve. Place the mouse pointer on the graph. Press the **right** mouse button and drag the mouse pointer while holding down the right button. The graph is redrawn continously while the mouse is moved.

Trace and sample details

Click on the link "Show Details" in the lower right corner to show detailed information about the trace and a specific point on the curve.



Under the heading **Selected Trace**, information about the currently selected trace is displayed. When several traces are displayed (using the Select option to open traces from TC) you can use the buttons ▲ and ▼ to browse through all traces. Note that two curves can be displayed for each trace.

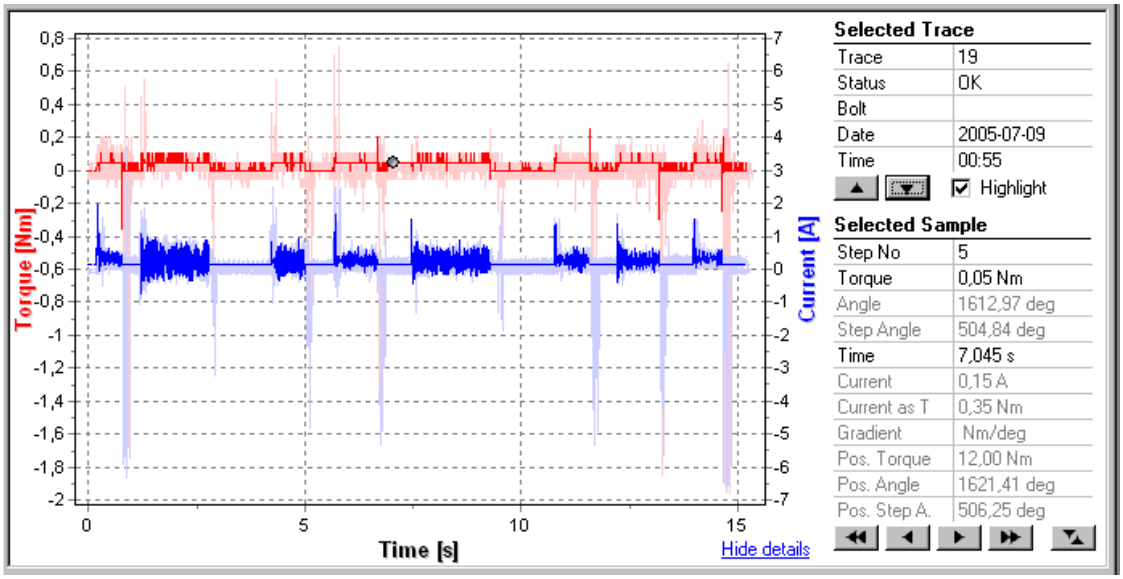
In the identity list (displayed by clicking on the button **Identity>>**) more information can be found for the trace.

Under the heading **Selected Sample**, information about the currently selected sample is displayed. To select a sample, click with the mouse on the graph. The point on a curve that is nearest to the mouse pointer is selected and marked with a black dot.

The information that belongs to the actual curve is displayed with black text. Other information about the selected sample is displayed with gray text.

Use the buttons to fine tune which sample that is selected. The buttons ◀◀ and ▶▶ moves the point 10 samples backwards or forwards in the buffer. The buttons ◀ and ▶ moves the point only 1 sample backwards or forwards. Use the button ▼▲ to toggle between the two quantity curves for the trace. In the picture above, clicking on this button would move the selected point from the **Angle over Time** curve to the **Torque over time** curve.

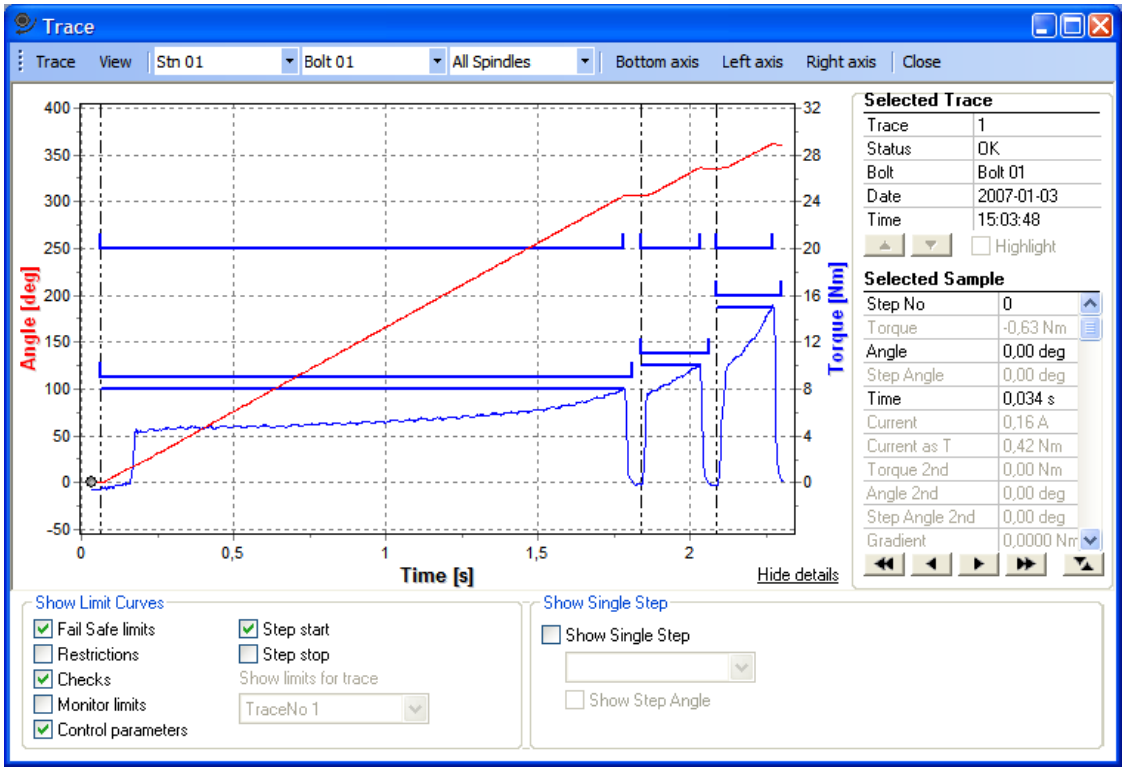
Use the checkbox **Highlight** if you want to study one specific trace among all the others.



The curves of the selected trace are highlighted compared to the other traces. This makes it easy to study a specific trace while still having several others displayed. Change the selected trace to highlight another trace. This makes it possible to browse through all traces viewing them one by one.

Show Tightening limits and parameters

If you select **Extras | Show** from the **View** menu the form is expanded and displays controls that are used to specify additional data to be displayed in the diagram.



The following additional data may be added to the diagram:

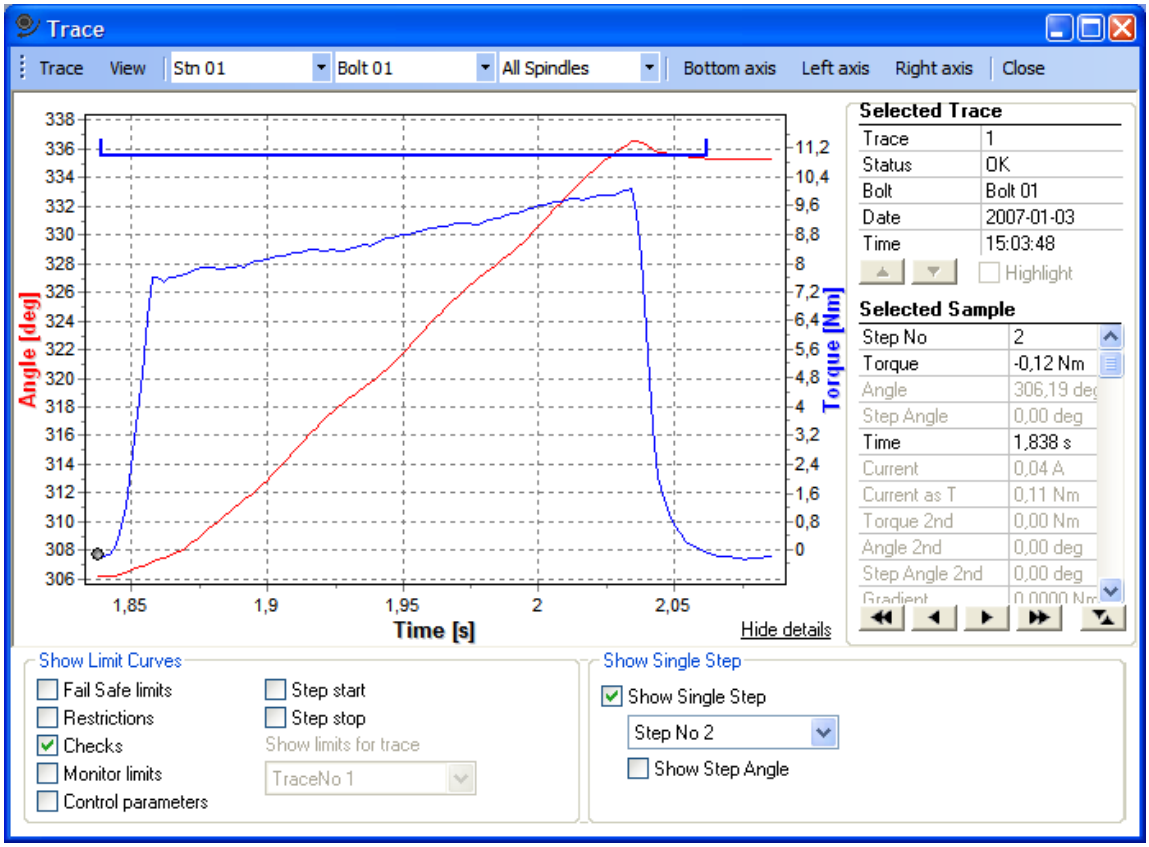
- **Fail Safe limits**
- **Restrictions**
- **Checks**
- **Monitor Limits**
- **Control Parameters**
- **Step Starts and Stops**

To include any of the groups above in the trace check the corresponding checkbox. The respective limits or parameters are displayed only if they are relevant with respect to the trace curves currently displayed. Torque limits are displayed when Torque over Time curves are viewed, angle limits are displayed when Angle over Time curves are viewed, monitoring limits and other checks that measure torque vs. angle are displayed when Torque over Angle curves are viewed. The descriptions of Step – Control, Step – Restriction, Step – Check and Bolt Monitoring describe if any limits are shown in trace and in what view they are shown.

Note that limits and parameters are only shown for one trace at the time. Use the combobox "Show Limits for trace" **or** use the buttons ▲ and ▼ under the Selected Trace heading to change the trace to show limits for.

Show Single Step

It is possible to view only one step at the time. Select **Extras | Show** from the **View** menu to bring forward the **Show Single Step** frame.



Mark the checkbox "Show Single Step" and select which step number to show in the combobox.

If quantity Angle is used on either a vertical axis or the horisontal axis, you can select to show the step angle instead of the cycle angle with the checkbox "Show Step Angle". The step angle starts at zero degrees at the beginnig of each step.

Note! The step is considered to begin at the step start time. The step angle is counted from this point also. This may not be the same position as other parts of the system use for the start of the step. The positions of limit curves may differ from what is expected.

Show Cycle Data

If you select **View | Extras | Show cycle data** the lower part of the window is expanded and displays more data. Press the **Layout** button to set up the layout of the cycle data. This will invoke the Trace Reporter, see chapter: Edit reporter.

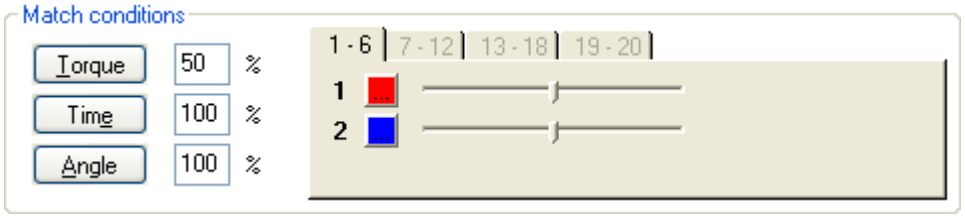
Reference trace

It is possible to save any displayed trace as a reference trace. The reference trace is stored by selecting **Trace | Save as reference** from the menu. There can only be one reference trace on ToolsTalk PowerMACS.

If a reference trace is stored it can be displayed in the background of all other traces by checking the **View | Reference trace** option. The reference trace is displayed with more transparent colors than normal.

Match curves

When displaying several traces at the same time there are many reasons for the traces to differ on the horisontal axis. Since this makes them difficult to compare it is possible to match the traces to each other. Select **View | Extras | Match conditions** to display a frame where you can set up the match conditions.



If you select **Torque,** matching will be done where the traces have reached the same level of torque. The level is specified in % in the field next to the alternative. If you e.g. specify 50% the traces will be matched to where the individual traces have reached 50% of their maximum torque (for the first time).

For **Time** and **Angle** it is probably more interesting to set 100%, i.e. where they have reached their final angle or time.

The scale of the horisontal axis will be set for the latest trace.

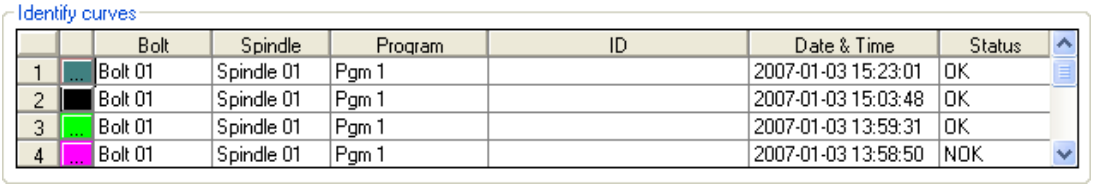
Use the sliders for minor positioning sideways.

With the buttons next to the sliders you can change color of the traces. **Note** that when two quantities on the vertical axis are displayed, it is **not** possible to change the colors. You have to select one of the axis to be *Nothing*.

By placing the mouse pointer over the button or the slider, a tool tip text is shown with identity information about the trace. You can change the color by clicking on the color identifiers next to each slider.

Identity of traces

Select **View | Extras | Identify** from the menu to display a list with identity information about each trace.


Identify curves

		Bolt	Spindle	Program	ID	Date & Time	Status
1	...	Bolt 01	Spindle 01	Pgm 1		2007-01-03 15:23:01	OK
2		Bolt 01	Spindle 01	Pgm 1		2007-01-03 15:03:48	OK
3	...	Bolt 01	Spindle 01	Pgm 1		2007-01-03 13:59:31	OK
4	...	Bolt 01	Spindle 01	Pgm 1		2007-01-03 13:58:50	NOK

It is possible to change color of one or many traces at the same time. To change color of one trace, simply click on the colored button to the left in the list. A color dialog is displayed where a new color can be selected for the trace. To change color of many traces at the same time, hold down the CTRL key and click on the traces to select them. Then click on one of the selected traces colored button to the left.

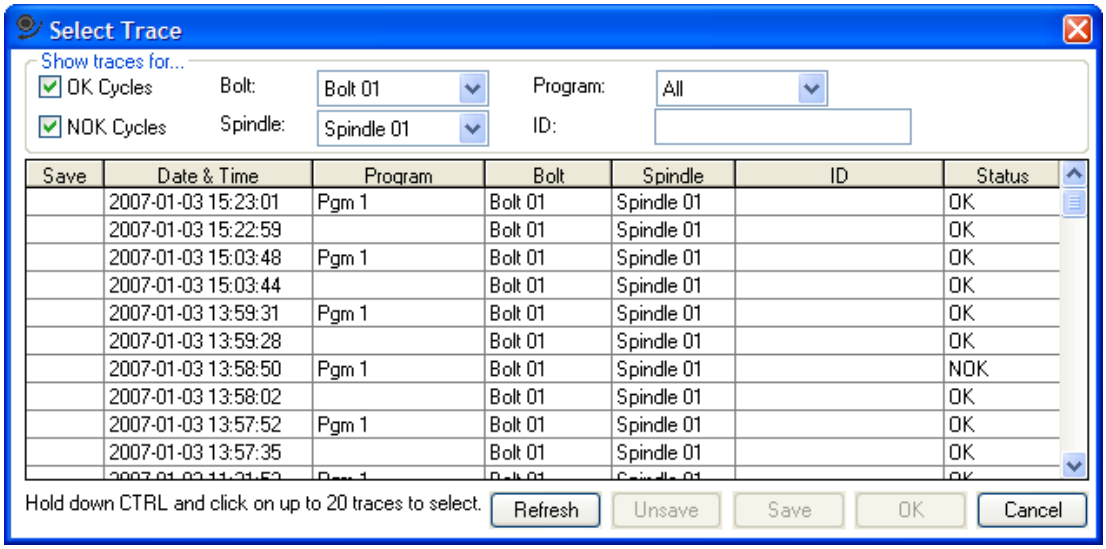
Note! It is only possible to change the color when only one curve is displayed for each trace, i.e. only one quantity on the vertical axis is selected. If two quantities are selected, all curves for the quantity on the left vertical axis will be displayed in red and all curves for the quantity on the right vertical axis will be displayed in blue.

Print

Select **Trace | Print** from the menu to print the trace currently displayed.

3.6.5.3 Select Trace

This dialogue is invoked from the View Trace form by selecting the **Trace | Select from TC** menu option and is used for selecting one or several traces to display.



The list shows traces stored. By use of the **Show traces for…** frame you can select which traces to show. In the ID field write as many characters you like. All traces that have an ID that contains the string of characters will be displayed.

Hold down the CTRL key and select with the mouse the traces you are interested in. When you return to the view window all the traces are presented in the same diagram, with different colors.

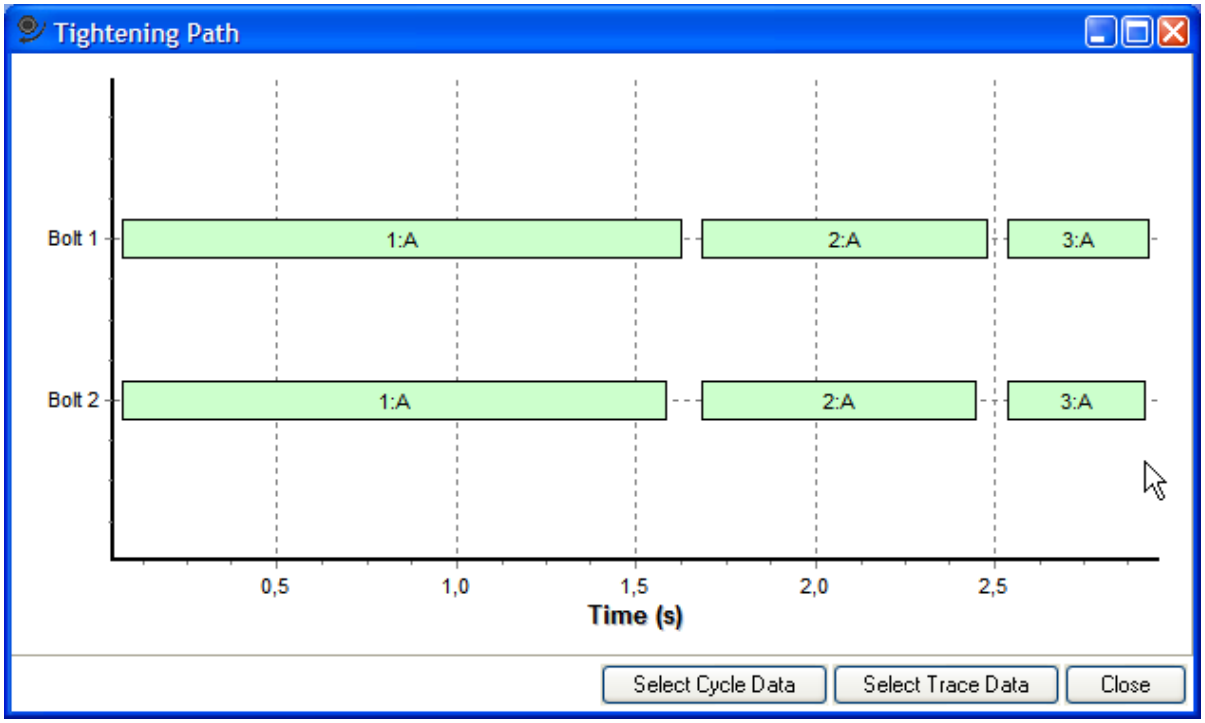
Normally traces are from the most recent cycles. Traces, that are tagged for save, will be saved indefinitely. The save tag can be switched on and off by use of the **Save/Unsave** buttons for any trace. Saved traces are marked by an S in the Save column on the left side of the window.

Due to the fact that traces consume quite a lot of memory only a limited number of traces can be saved. When trace memory is full and new traces are produced the system will strive to keep the number of stored OK and NOK according to the SRAM configuration (see chapter SRAM). This means that a new OK trace will overwrite the oldest OK trace not marked for save, and vice versa for NOK traces.

Note! All traces, also the ones marked for “Save”, are deleted if the battery backed up memory is cleared, for example when new System Software is loaded.

3.6.6 View Tightening Path

The Tightening Path window displays how one or more bolts executed a cycle as a gantt chart. This layout makes it more easy to see how reject management procedures affected a cycle.



To display the tightening path for a cycle, either select a cycle data by pressing the Select Cycle Data button or select one or more traces to compare.

In order to have the step type written for each step in the graph the Step Type cycle data variable must be present in the Trace reporter.

3.6.7 View Event Log

In the PowerMACS system events could happen that the operator should be informed of. These can be of following types:

Event type	Num. Id	Example
General	0	Events that do not belong to any of the specific categories below.
Access	1	Operator log-in and log-out, failed log-in attempts
Power up	2	Restart of system
Emergency stop	3	Emergency or Machine stop of station
Checks	4	Tightening application errors, ”Torque high”, ”Angle low” etc.
System error	5	Abnormal situation, hardware breakdown etc.
Modifications	6	Changed parameter. Event contains additional information, like parameter, station, spindle, program etc.
Hardware error	7	Hardware related errors, servo failure etc.
Software error	8	Internal communication errors, etc.
Ext. com. error	9	No response, to many retransmissions, etc., when communicating with external equipment.
SPC	10	SPC events, "7 up", "7 down", etc.
Setup	11	Erroneous configuration detected during run mode, e.g. missing parameters, etc.

All events are also classified by their Severity. Severity is one of the following:

Severity	Num. Id	Example
Info	0	Events that only informs the user of some activity, for example "Power on".
Warning	1	Events that may indicate unexpected behavior, for example "Machine stop".
Error	2	Severe errors in the system or events that indicate that cycle failed, the error might be repairable.

Num. Id. above is the numerical identity of the respective Event Type. This value is used when events are reported to binary devices. See Layout of Events for a complete description of the binary format of events.

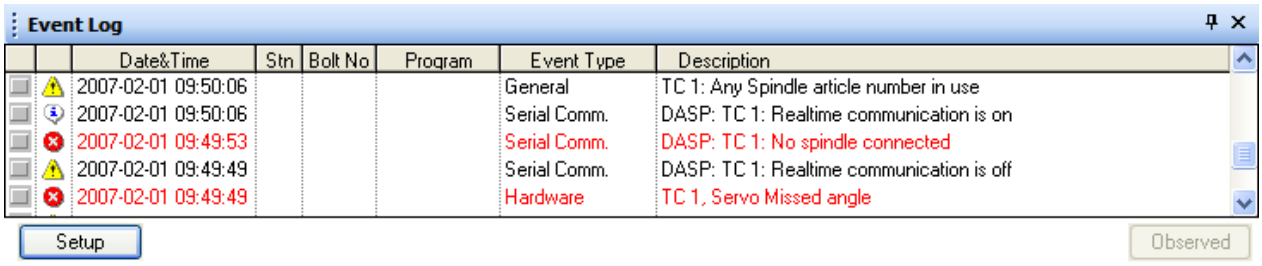
All events are stored in an Event Log, which can be viewed using ToolsTalk PowerMACS.

For each event the following information can be displayed (from left to right):

- If the event is observed or not (a check mark if observed)
- The severity of the event
- The date and time when the event occurred
- Number of the TC that generated the event (Optional)
- The number of the station to which the event belongs (zero (0) if not connected to a particular station) (Optional)
- The number of the bolt (the user specified bolt number) to which the event belongs zero (0) if not connected to a particular bolt) (Optional)
- The Angle Channel if relevant (for restriction, check and monitoring events). Possible values are:
  - M Monitoring
  - C Control
  - 1,2 Channel number

  (Optional)
- The Torque Channel if relevant (for restrictions, check and monitoring events). Possible values are:
  - M Monitoring
  - C Control
  - 1,2,3 Channel Number (1-2 is torque transducers, 3 is Current As Torque)

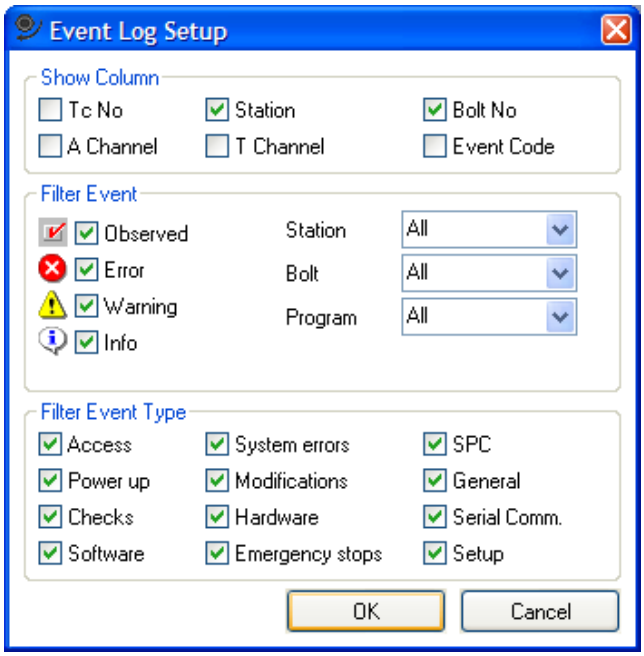
  (Optional)
- The name of the program to which the event belongs (blank if not connected to a particular program)
- The type of the event
- The event code of the event (Optional)
- A textual description of the event



By default the Event Log window is docked on the bottom right half of the ToolsTalk PowerMACS main window but can be placed freely.

Events that have been dealt with can be marked as “Observed”. Select one or more events in the list and press the **Observed** button. A single event can be marked by just double clicking it. An observed event shows a red check mark in the leftmost column.

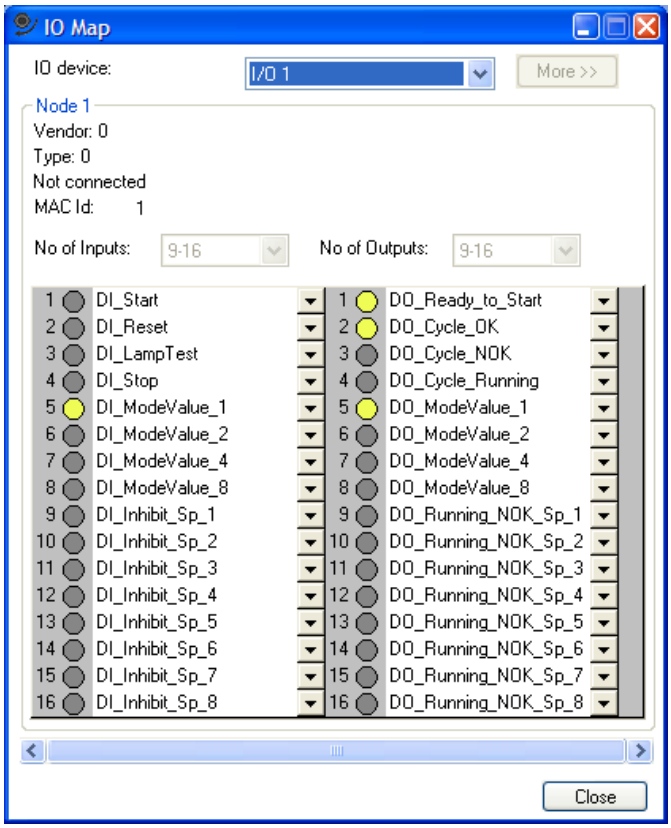
If you press the **Setup** button the **Event Log Setup** window is opened where you can specify which event types and/or Severities to display. You can also include just events connected to a certain station, bolt, or program.



Events may also be viewed and marked as observed per event type from PowerMACS PLC. See the description of PLC Event handling in chapter: Advanced Station Settings for how this is enabled.

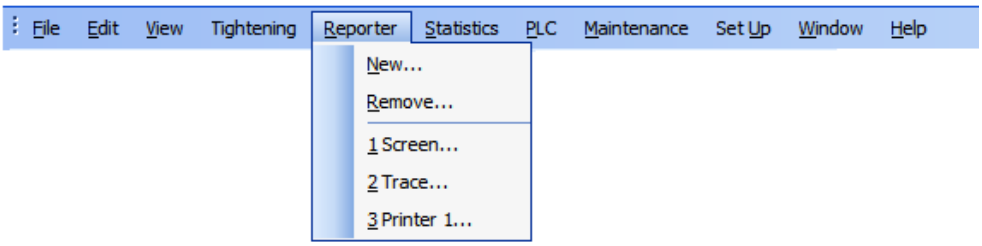
3.6.8 View I/O signals

When ToolsTalk PowerMACS is connected you can display the current status of all digital inputs and output connected to a particular TC. Use menu choice **View-IO** to open the **IO Map** form. (You can also open it from the Hardware on the **System Map** form. Select the IO device for a TC and press **Advanced...**.)



3.7 Reporter

The reporter is a very flexible and powerful tool that is used to configure which data to display and output from a PowerMACS system. Reporters can be added, edited and removed from either the **Reporter** menu item or the system map.



It can be seen as a function that helps you select various data from the system. It controls what data to send to a particular device and how this data is formatted. The reporter itself is not able to output any data; the device that the reporter is connected to does this.

Some devices, such as the ToolsTalk PowerMACS Screen, the ToolsTalk PowerMACS Printer, the ToolsTalk PowerMACS File, and the Trace are considered to be logical devices that always exist, meaning that you do not need to create reporters for these.

To output data via any other type of device, the device must first be added to the PowerMACS system. See Add a device for how to do this. When the device exists you may create a new reporter, connect it to the device and finally use the reporter to set up what data to report.

3.7.1 Predefined reporter settings

Files that describe predefined reporters are installed together with ToolsTalk PowerMACS. The files are placed in special directories depending of the type of reporter. For example, API reporters are placed in the directory "<ToolsTalk PowerMACS install dir>\Reporters\API" and screen reporters are placed in the directory "<ToolsTalk PowerMACS install dir>\Reporters\Screen".

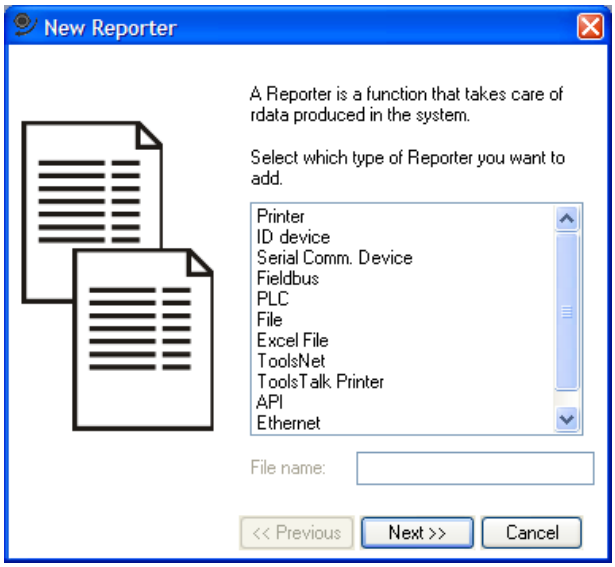
It is possible for users to make their own reporter files and place them in this directory structure. To do this, simple export a reporter by using the **Export Table** form (invoked using the **File - Export** menu item, see **Table Export and Import**). Place the exported file in the special directory for the Reporter type that you want to use it for.

When adding a device in the File-New Wizard, the user is presented with a list of predefined reporters to choose between. This list is built up by taking all file names of the files found in the special directory for the type of device in question. The chosen reporter file will then be imported to the setup and used for the device.

The list of predefined reporters is also available when adding a device from the System Map window. After adding the device, the user gets the question if he wants to add a reporter. If the answer is yes, the list described above is presented. Note that it is only reporters for the specific type of device that are displayed. It is also possible to browse for and select a predefined reporter from the Reporters window.

3.7.2 New reporter

To create a new reporter use menu item **Reporter-New…** or right click on the **Reporters** section of the System Map and select **Add** Reporter. This will display a wizard that guides you through the creation of the reporter.



From the list select which type of reporter you want to create:

Type	**Used for configuration of data…**
ToolsTalk Printer	to be printed on a printer using ToolsTalk PowerMACS (requires ToolsTalk PowerMACS to be connected)
File	sent to a file via ToolsTalk PowerMACS (requires ToolsTalk PowerMACS to be connected)
Excel File	sent to an Excel file via ToolsTalk PowerMACS (requires ToolsTalk PowerMACS to be connected).
Printer	to be printed on a (line) printer directly connected to a TC
PLC	sent to inputs of PowerMACS internal PLC (defined in the CycleData_Var section of the Global_Variables worksheet)
ID device	sent to ID devices of type escort memory that are connected to the TC
Ext. Comm. Device	available using external communication devices
Fieldbus	available using fieldbus devices
API	available using the PowerMACS API (Application Programmer Interface) for access by other PC based computer applications via the network
ToolsNet	sent to a Atlas Copco ToolsNet server

You cannot create (or delete) reporters for the Screen, that is the ToolsTalk PowerMACS Cycle Data window, or the Trace, that is the cycle data display in the Trace window. On the other hand, these reporters are always available in the system.

If wanted, specify an additional name of the reporter. If you create a reporter of type Printer and you add the name HP6P you can later find a menu item **Reporter - Printer HP6P** that corresponds to this reporter.

The additional name specified for a **File** or **Excel File** reporter is used as the name of the result file. For each File reporter two files are created. The first have the character “1” appended to its name and the second, which is created when the first file is full, the character “2”. When the second file is full then the first will be overwritten. All files are created in the directory **Log** located in the directory where you to install ToolsTalk PowerMACS.

The **Excel File** reporter can only be used if Microsoft Excel is installed on the PC that ToolsTalk PowerMACS is running on. If Excel cannot be started the data is written to a text file with extension *.txt instead. This text file is formatted in a format suitable for Excel. To import the text file from Excel follow do as follows:

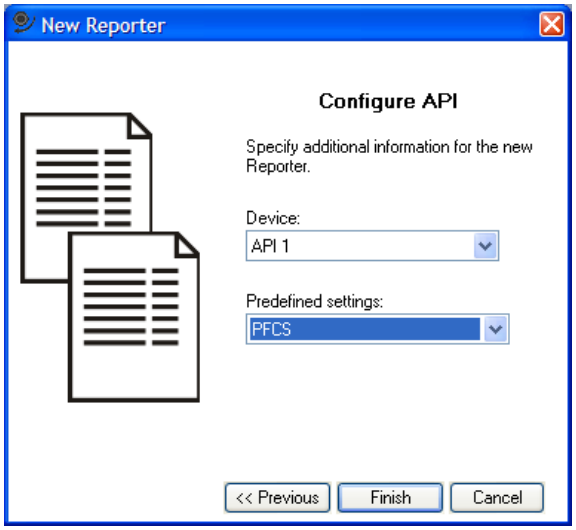
- From inside Excel choose File/Open.
- In the Open dialogue box select "Text files (*prn; *.txt; *.csv)" as "Files of types". Then select your file and press Open.
- In the Text Import Wizard - Step 1: Select "Delimited" in the frame marked "Original dat type" and press Next.
- In the Text Import Wizard - Step 2: Check "Tab" in the frame "Delimiters" and press "Finish".

Please note that if the destination Excel file is locked by another user ToolsTalk PowerMACS will fail to write data to it. In these cases ToolsTalk PowerMACS will create a new file named

Name_DataWhenFileWasLocked_YYYYMMDD-HHMMSS".xls

where "Name" is the original name assigned to file and "YYYYMMDD-HHMMSS" is the current date and time (example: "Excel1_DataWhenFileWasLocked_20020514-125932.xls"). The temporary file will be used until the next time you go off-line.

When ready with the first page then press **Next >>** to display the next page of the wizard. If needed, depending on the type of reporter, you will be asked for some additional information. For an API reporter it looks like this:



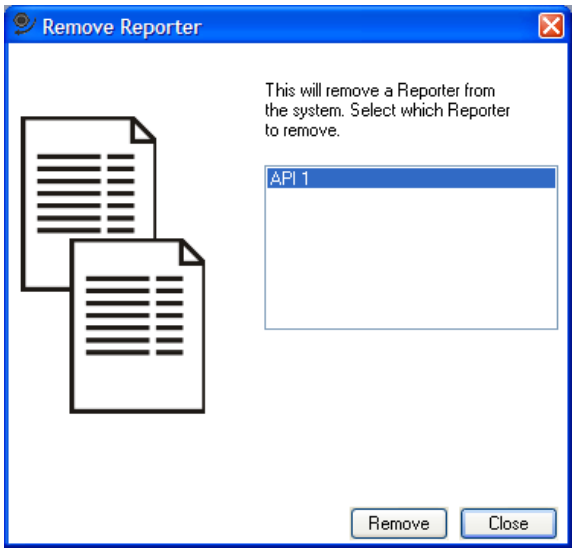
For most of the types you must select which device this reporter should connected to. The Device list box displays all devices, of the relevant type, existing in the system. See **Add a device** for how to add new devices if you need to that.

If predefined reporter settings files exists in the directory for the currently selected reporter type (see **Predefined reporter settings**), you have the option to select predefined settings for the new reporter. If you do not want this, leave the selection blank, which will create the reporter without any settings.

When ready, press **Finish**. A new window is displayed as if you had used the menu item **Reporter-Open…** In this specify what data to collect and the layout.

3.7.3 Remove reporter

You can remove a reporter by menu item Reporter-Remove…



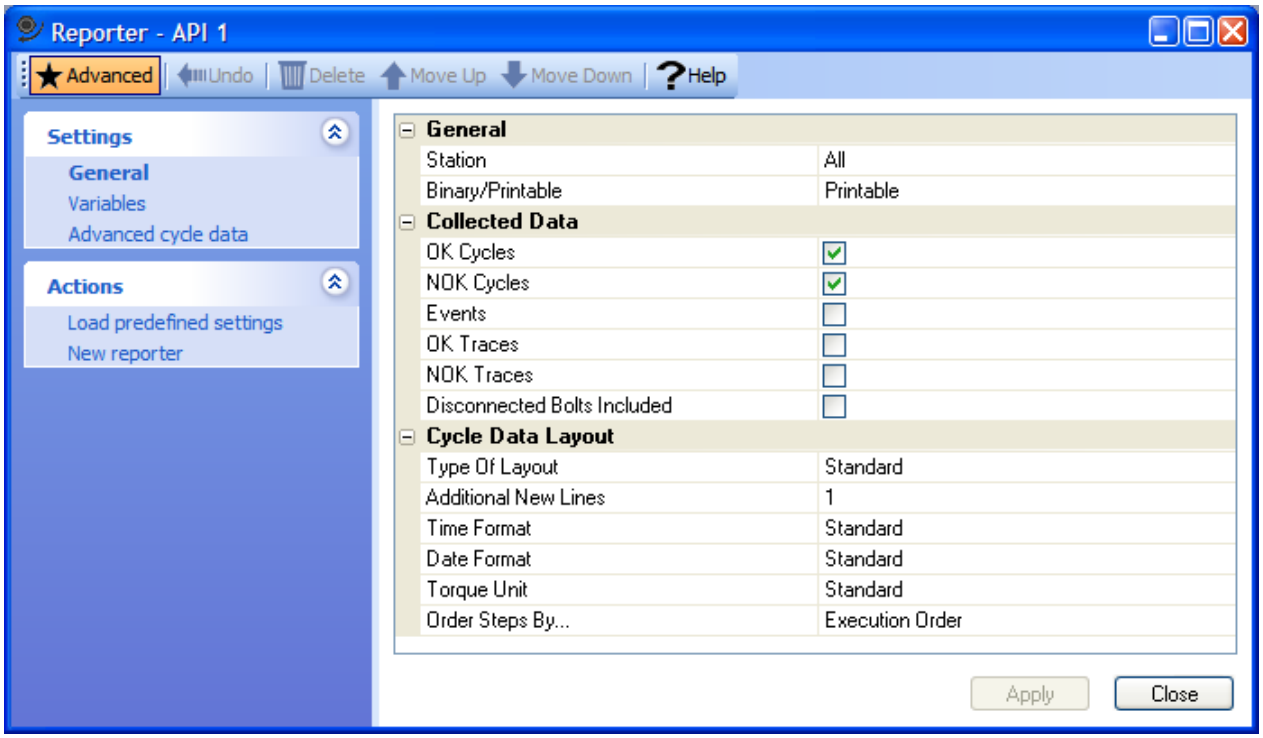
Select a reporter and press **Remove**.

You can also delete reporters by right-clicking on them in the system map and select Remove Reporter.

3.7.4 Edit reporter

The reporter is used to specify what data to collect for a device and the layout of the data when presented on the device. All devices that should handle process data; Cycle Data, Events and Traces, must have a reporter set up for it in order for any data to be output.

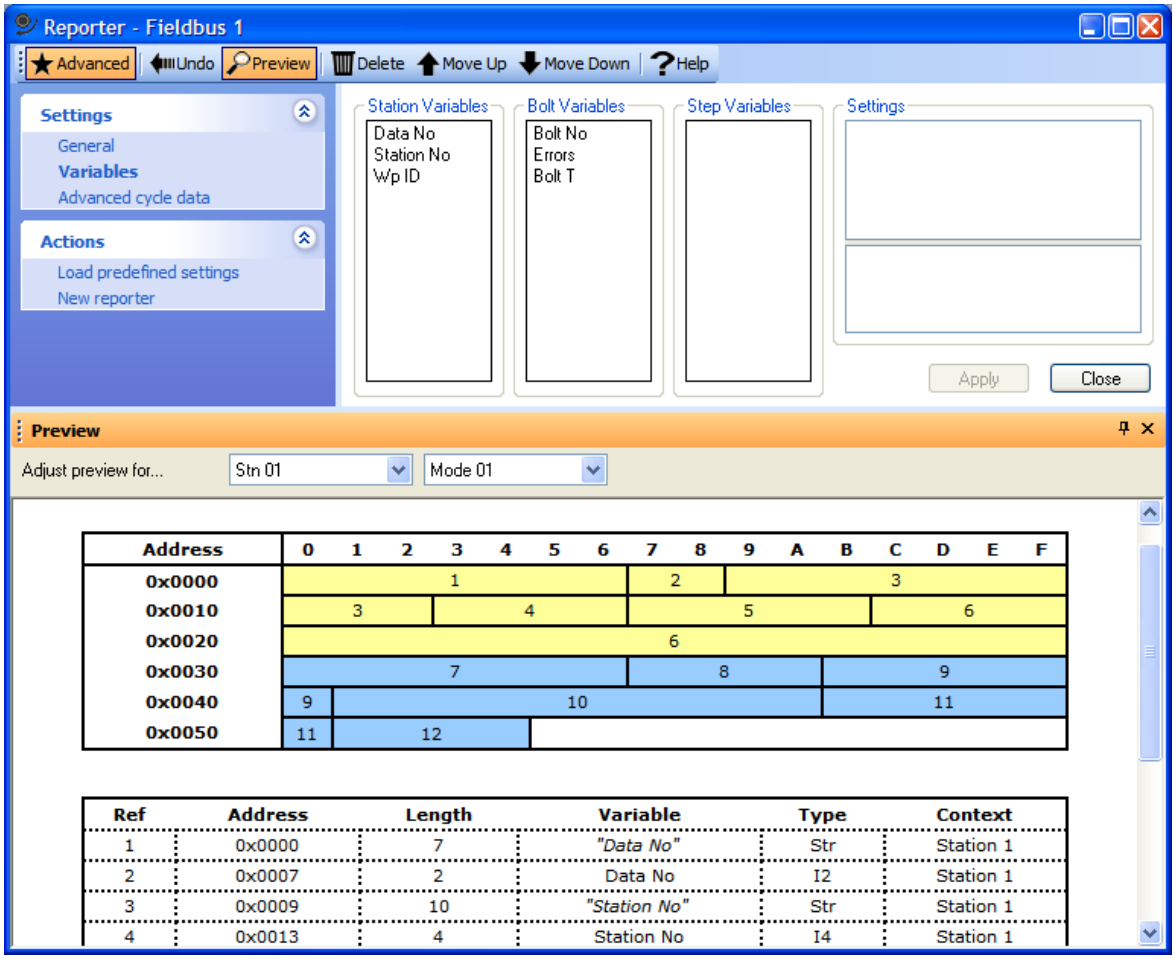
To edit a reporter select its name in the Reporter menu, e.g. **Reporter – Printer 1...** or double click on it in the system map. This will open the following form:



Specify in the **Station** field, which specific station you want data from, or **All** stations if data generated by all stations should go to the device. Use the **Binary/Printable** combo box to select if the data should be formatted as binary or printable output.

The controls grouped under **Collected data** are used to configure what type of Process Data that should be accessible for the device. Use them as follows:

- Select if **OK Cycles** should be reported and for certain reporters the color to print them.
- Select if **NOK Cycles** should be reported and for certain reporters the color to print them.
- Select if **Events** should be included and optionally the color used for printing. More specific filtering may be defined using the **Reporter Preview**
- For binary reporters it is possible to preview how the generated data will be formatted.



It is possible to adjust the preview according to a station and mode number. It is possible to hoover with the mouse above the individual fields in the address map to view the actual variable.

- Advanced Event settings form.
- Check **OK Traces** to include all traces with bolt status OK (only reporters capable of traces)
- Check **NOK Traces** to include all traces with bolt status NOK (only reporters capable of traces)
- Check **Disconnected bolt included** if you want such bolts to be included in the output.

For presentation on devices that can handle colors (the Screen and ToolsTalk Printer), you can also specify the color you want the data to have. You may for example have events and NOK cycles shown in red. (Traces can only be output in binary form therefore color is not relevant.)

For reporters of type **File** or **Excel File** the size, in kByte, of the resulting file is configured with the parameter **File Size**.

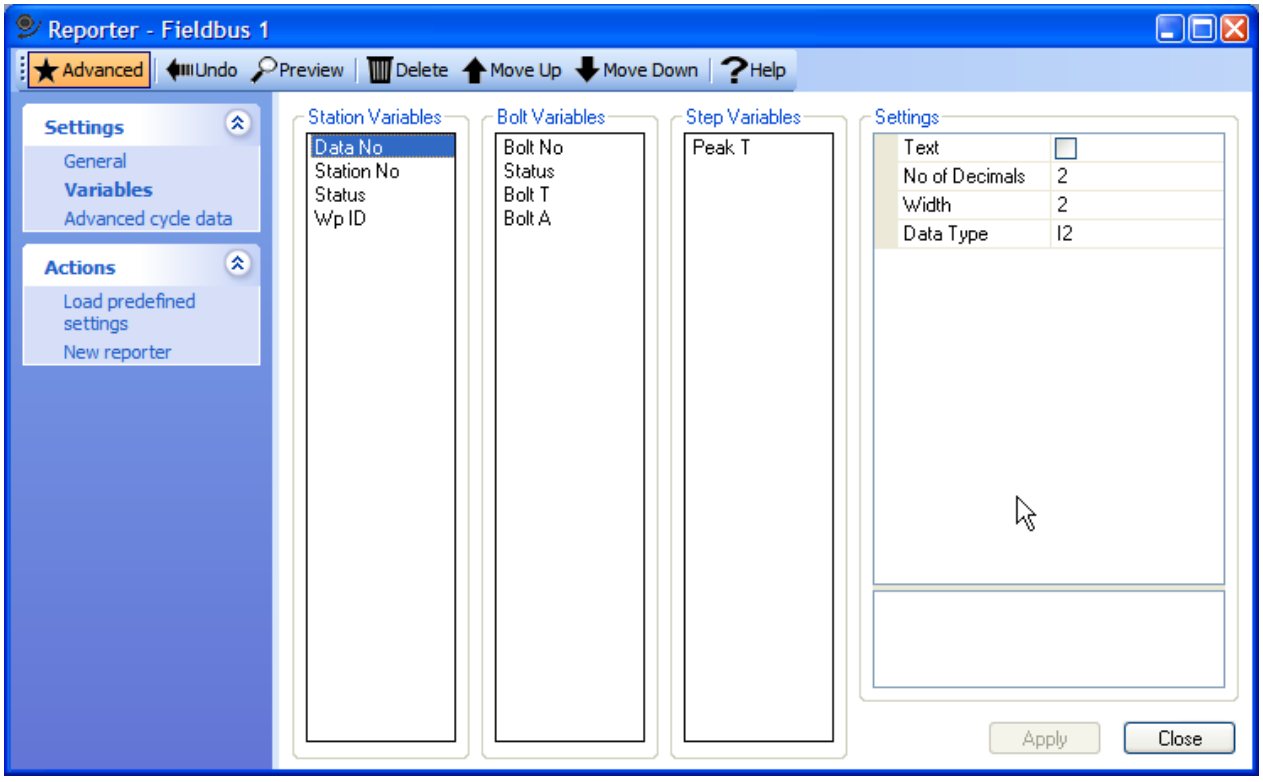
For the Screen, ToolsTalk Printer and Trace reporters you can specify which font to use. Please keep in mind that you must use a font with fix width to make the variables be printed in columns (for example when printing cycle data as table as described in chapter: Adding Cycle Data Variables.

Select Variables on the left side of the Reporter window to show the variables to include in the cycle data.

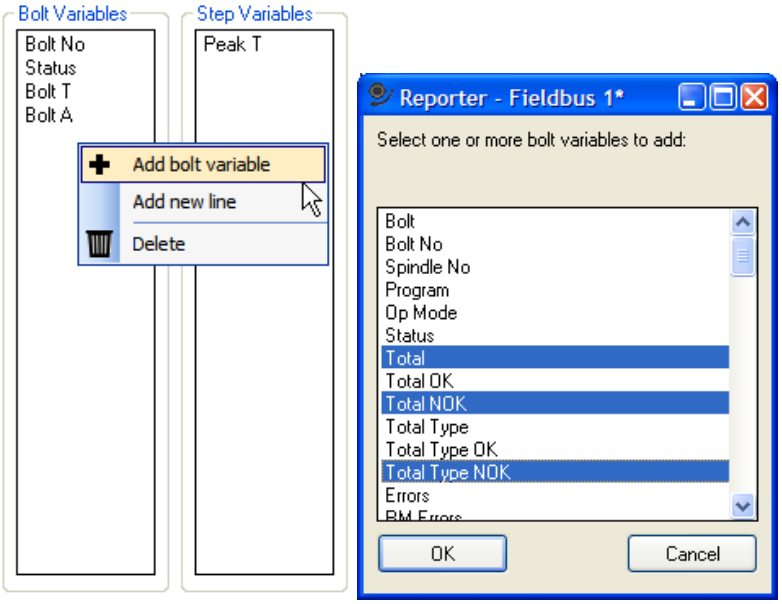
It is possible to select a predefined reporter by clicking the Load predefined settings… located under Actions on the left side of the reporter window. The list with predefined reporters of the same type as the currently configured reporter is displayed (see Predefined reporter settings), but it is also possible to browse freely for any predefined reporter settings. If the chosen predefined reporter settings contain parameters that are not available or configurable for the currently configured reporter the parameters are omitted and fix parameters remain unchanged. An example is the trace reporter, which can never contain station status. If predefined settings containing station status are applied station status will still not be configured for the trace reporter.

3.7.4.1 Adding Cycle Data Variables

Select **Variables** on the left side of the Reporter window to show the variables to include in the cycle data.



Three list boxes with **Station**, **Bolt** and **Step** variables are presented here. To add a variable simply right click on any of the listboxes and select **Add variable** from the menu – a list with possible variables will be presented. It is possible to hold Ctrl and select more variables at the same time.



It is also possible to drag & drop variables from the **Dynamic Tool** to the three variable list boxes.

The **Settings** property grid shows the settings for the currently selected variable.

Setting	Usage
Text	Check this checkbox to include the variable name as a leading, or prompter, text. When using table layout (Header text frequency > 0) then the variable name is always used as column header regardless of the selection in this column.
No Of Decimals	Specifies the number of decimals to print. Only valid for variables of numerical type and when format is Printable (Data/Printable selected).
Width	Defines the width of the field, in number of characters, in which the **value** of the variable is printed. (left adjusted). When using table layout (Header text frequency > 0) then the width must be big enough to cover the variable name since it is used as column header. When format is binary (Data/Binary selected) then Width normally should be left blank for all numerical values variables. This will cause the selection in the Type column to control the width of the field. However, specifying a value that is larger than the number of bytes implicated by the Type selection will make the variable occupy the specified number of bytes. Data is then left adjusted (the field is padded with NULL (0) characters).

Type	Controls the data type of the variable when printed. Valid only when format is binary (Data/Binary selected). Takes the following values • I1 - Value is formatted as a one-byte integer. Any fractional part is truncated. • I2 - Data is formatted as a two-byte integer. Any fractional part is truncated. • I4 - Data is formatted as a four-byte integer. Any fractional part is truncated. • I8 and I12 - Data is formatted as 8 or 12-byte error/warning information. Only valid for the bolt level result variables "Error", "RM Error" and "Warning". See also chapter: Errors and Warnings. • F - Data is formatted as a four-byte Float (either as IEEE754, Fixed 2I2D, SPECIAL 1 or SPECIAL 2). • Str - Data is formatted as a string. The setting of Width controls the number of bytes occupied.
Step Numbers	Specifies for which steps a variable will be printed. Only valid and visible for Step data variables. Enter the step number of the step(s) for which the variables should be included. Leaving the value blank will print the variable for all steps.

Note! Selecting a variable in the reporter will cause the variable to be included in the result. However, *this does not automatically mean that the variable will be measured*. Most of the variables correspond to values measured by bolt monitoring and step checks. Whether these checks are performed or not is controlled by the Tightening program being executed. See Bolt Monitoring and Step – Check for how to set up these.

3.7.4.2 Layout of Cycle Data

The controls grouped under **Cycle Data Layout** specify how the cycle data results should be presented.

The variables that can be included in the result are divided in three sections, **Station data**, **Bolt data** and **Step data**. Station data variables are reported once for each cycle, Bolt Data variables once for each bolt and Step data variables for each step in the program.

Normally the result is always laid out in a hierarchically manner, with the Station data at the top and the Step data at the bottom, as below:

```
Station data
   Bolt data for bolt 1
      Step data for bolt 1, step 1
      Step data for bolt 1, step 2
                  :
      Step data for bolt 1, step n
   Bolt data for bolt 2
      Step data for bolt 2, step 1
      Step data for bolt 2, step 2
                  :
      Step data for bolt 2, step n
                  :
```

Variables that do not have a valid value, for example due to that the check that produces the value is not executed or data from disconnected bolts, are said to have the value NOT_DEFINED. These are printed

as “blank” (all spaces) when **Data** is set to **Printable**. With **Data** set to **Binary** all undefined numerical variables are set to the value –32768 while text variables are set to an empty string, that is “”.

For a description of all variables see chapter: Result variables.

The parameter **Type of layout** controls how the printed result is formatted. Select one of the following alternatives:

- **Standard**. This generates a layout where the variables are formatted exactly as specified on the **Variables** view (see Adding Cycle Data Variables).
- **Table**. Formats the result as a table, that is, the variables are printed in columns with their respective prompter texts as column headers.
- **Standard + Table**. This layout is a mix of the previous ones where the station header is formatted using the "Standard" layout, while bolt and step level result are formatted using the "Table" layout.
- **Fixed Positions** This option gives you the possibility to have each of the variables at a fix position in the result data. The output is similar to the Standard option with the following exceptions.

  First the bolt level result variables are primarily ordered as defined in the variable view and then in bolt number order. That is, the first bolt level result variable is printed for all bolts, then the second bolt variable for all bolts.

  Note that the bolts are ordered using the internal, ordinal, bolt number and not using the user defined bolt number. A particular variable is always printed at the same position for a given bolt regardless if the cycle result contains data for all bolts or only a few. Unused positions are filled with zeros (0) if binary output selected and spaces if printable output is chosen. Should a cycle result contain several data for the same bolt the most recently produced data is used.

  Secondly, all step level result variables are disabled and cannot be included in the report.

  Station level result variables are printed just as for the Standard option and all format options can be used.

  Since several stations in a system can use the same reporter the number of bolts per cycle data can vary also if all bolts in the station are run. It is therefore possible to configure how many bolts, or actually bolt positions, that should be used when the bolt result variables are laid out. The **Fixed pos. Layout** parameter **Report bolts** on Advanced CD settings controls this.

  This option is only available for the TC based reporters Fieldbus, Printer, API, ID device, Ext. Comm., and PLC.
- **Static Offset**. This format is similar to the “Standard” format but allows for offset specification of where bolt and step data should begin. Configurations of offsets are displayed below **Cycle Data Layout** when **Static Offset** is selected.

  This format type makes it possible to divide a cycle data result into several blocks with areas i between where no data is written. Note that the offsets specified are relative to the start address

of the targeted device output area.

It is possible to select where bolt data should start after the station data and how much each bolt should occupy. Within each bolt it is possible to specify the same details for step data.

See Cycle Data layout example for examples of the different formats.

The parameter **Header frequency** controls when a header row is printed when the **Layout** option **Table** or **Std. + Table** is used. The value 10 of Header frequency will generate a header for every 10th cycle data printed.

Enter in the **Additional new lines** field the number of empty lines to print after that all cycle data has been printed.

Using the **Torque unit** field you can specify the unit in which all torque results are to be presented. The system will recalculate values before presentation. Set Standard if you want to keep the basic torque unit set using the **Options** form invoked from the **Set Up** menu.

Using the **Date format** and **Time format** parameters you control how date/time variables should be formatted when printed. Select "Standard" if you want to keep the format specified on the **Options** form. For the ToolsTalk PowerMACS based reporters Screen, Trace, File, ExcelFile and ToolsTalk Printer, the alternative "Reg. Settings" is available. Chose it if you want to use the date/time format configured using Regional Options on your PC.

The effect of the selected date and time formats is displayed in the status bar at the bottom of the Reporter form.

For reporters configured to format the result binary (Data/Binary selected) you can also:

- Control byte order of all numerical variables. Set **Byte Order** to **Normal** to have the most significant byte printed first, or at the lowest address. This is also known as Big Endian or Motorola format. Select **Intel** to have the least significant byte printed first (Little Endian).
- Control how real values are formatted. Set **Float format** to **IEEE754** to print them as standard 32 bit floating point values according to IEEE 754.

  Choose **I2D2** to have them printed as a fixed-point value where the two most significant bytes contains the integer part and the two least significant bytes the decimal part multiplied with 10000.

  For the altenatives **SPECIAL 1** and **SPECIAL 2** the real value is first multiplied with 10 and then rounded to an integral number before it is written. **SPECIAL 1** prints the resulting value as a Mitsubishi BCD value consisting of four consecutive bytes where the value 27.3 is written as 0x03, 0x07, 0x02, 0x00 with the first byte (0x03) at the lowest address. **SPECIAL 2** prints the resulting value as four ASCII decimal digits, that is, the value 27.3 is written as four bytes having the values 0x30, 0x32, 0x37, 0x33 ("0273") with the first byte (0x30) at the lowest address. Please note that the value NOT_DEFINED is reported as the real value 999.9.
- Control how the Station and Bolt level Status variables are reported using **Status format**. Choose **NORMAL** to have them reported with values according to chapter: Statuses.

Choose **1/0 (OK=1)** to have an OK result, that is, OK and OKR, reported with value 1 (one) and a NOK result, that is, NOK, NOKRM, and TERMNOK, reported with value 0 (zero).

Choose **0/1 ASCII (OK=0)** to have the status using ASCII digits. An OK result is then reported as the digit "0" (0x30) while a NOK result is reported as the digit "1" (0x31).

For all TC based devices you can control the order in which the step results are printed. This is done using the **Order steps by** combo box on the Step data tab. Select **Execution order** to print the steps in the order that they are executed and **Step number** to have them ordered in ascending step number order.

3.7.4.3 Cycle Data layout example

Standard layout

Configuring a reporter of type Printer with **Type of layout** set to "Standard", **Additional new lines** to 1 and the following result variables:

Station variables:

Variable	Width	No Of Decimals	Text
Station	8		Yes
Time	24		Yes
Status	9		Yes
<New Line>			

Bolt variables:

Variable	Width	No Of Decimals	Text
Bolt No	8		Yes
Bolt T	7	1	Yes
Bolt A	7		Yes
Status	7		Yes
<New Line>			

Step variables:

Variable	Width	No Of Decimals	Text	Step Numbers
Step No	8		yes	
Peak T	7	1	yes	
A	3		yes	
<New Line>				

Will produce the following print out:

```
Station: Stn 01  Time: 011030 09:40:58         Status: OK
Bolt No: 1       Bolt T: 15.1   Bolt A: 435    Status: OK
Step No: 1       Peak T: 2.4    A: 125
Step No: 2       Peak T: 15.2   A: 310
Bolt No: 2       Bolt T: 2.5    Bolt A: 436    Status: OK
Step No: 1       Peak T: 2.5    A: 124
Step No: 2       Peak T: 15.2   A: 312

Station: Stn 01  Time: 011030 09:41:10         Status: OK
Bolt No: 1       Bolt T: 15.1   Bolt A: 435    Status: OK
Step No: 1       Peak T: 2.4    A: 125
Step No: 2       Peak T: 15.2   A: 310
Bolt No: 2       Bolt T: 2.5    Bolt A: 436    Status: OK
Step No: 1       Peak T: 2.5    A: 124
Step No: 2       Peak T: 15.2   A: 312

Station: Stn 01  Time: 011030 09:40:22         Status: OK
Bolt No: 1       Bolt T: 15.1   Bolt A: 435    Status: OK
Step No: 1       Peak T: 2.4    A: 125
Step No: 2       Peak T: 15.2   A: 310
Bolt No: 2       Bolt T: 2.5    Bolt A: 436    Status: OK
Step No: 1       Peak T: 2.5    A: 124
Step No: 2       Peak T: 15.2   A: 312

Station: Stn 01  Time: 011030 09:41:44         Status: OK
Bolt No: 1       Bolt T: 15.1   Bolt A: 435    Status: OK
Step No: 1       Peak T: 2.4    A: 125
Step No: 2       Peak T: 15.2   A: 310
Bolt No: 2       Bolt T: 2.5    Bolt A: 436    Status: OK
Step No: 1       Peak T: 2.5    A: 124
Step No: 2       Peak T: 15.2   A: 312
```

Table layout

Using the same variable settings but with **Type of layout** set to "Table" and **Header frequency** to 2 gives the below printout:

```
Station Time                    Status   Bolt No Bolt T Bolt A Status Step No Peak T A
-------------------------------------------------------------------------------------
Stn 01  011030 09:40:58         OK       1       15.1   435    OK     1       2.4    125
                                                                      2       15.1   310
                                         2       15.2   436    OK     1       2.5    124
                                                                      2       15.2   312

Stn 01  011030 09:41:10         OK       1       15.1   435    OK     1       2.4    125
                                                                      2       15.1   310
                                         2       15.2   436    OK     1       2.5    124
                                                                      2       15.2   312

Station Time                    Status   Bolt No Bolt T Bolt A Status Step No Peak T A
-------------------------------------------------------------------------------------
Stn 01  011030 09:40:22         OK       1       15.1   435    OK     1       2.4    125
                                                                      2       15.1   310
                                         2       15.2   436    OK     1       2.5    124
                                                                      2       15.2   312

Stn 01  011030 09:41:44         OK       1       15.1   435    OK     1       2.4    125
                                                                      2       15.1   310
                                         2       15.2   436    OK     1       2.5    124
                                                                      2       15.2   312
```

Standard + Table layout

Using the same variable settings but with **Type of layout** set to "Standard + Table" and **Header frequency** to 2 gives the below printout:

```
Station: Stn 01  Time: 011030 09:40:58         Status: OK
Bolt No Bolt T Bolt A Status Step No Peak T A
-------------------------------------------------------
1       15.1   435    OK     1       2.4    125
                              2       15.1   310
2       15.2   436    OK     1       2.5    124
                              2       15.2   312

Station: Stn 01  Time: 011030 09:40:58         Status: OK
1       15.1   435    OK     1       2.4    125
                              2       15.1   310
2       15.2   436    OK     1       2.5    124
                              2       15.2   312

Station: Stn 01  Time: 011030 09:40:58         Status: OK
Bolt No Bolt T Bolt A Status Step No Peak T A
-------------------------------------------------------
1       15.1   435    OK     1       2.4    125
                              2       15.1   310
2       15.2   436    OK     1       2.5    124
                              2       15.2   312

Station: Stn 01  Time: 011030 09:40:58         Status: OK
1       15.1   435    OK     1       2.4    125
                              2       15.1   310
2       15.2   436    OK     1       2.5    124
                              2       15.2   312
```

Fixed Positions layout

Std. + TableSetting **Type of layout** to "Fixed Positions", the **Bolts to report** parameter to 5 (on the Advanced CD settings form) and the following variable settings:

Station variables:

Variable	Width	Dec	Text
Station	8		yes
Time	24		yes
Status	9		yes
<New Line>			

Bolt variables:

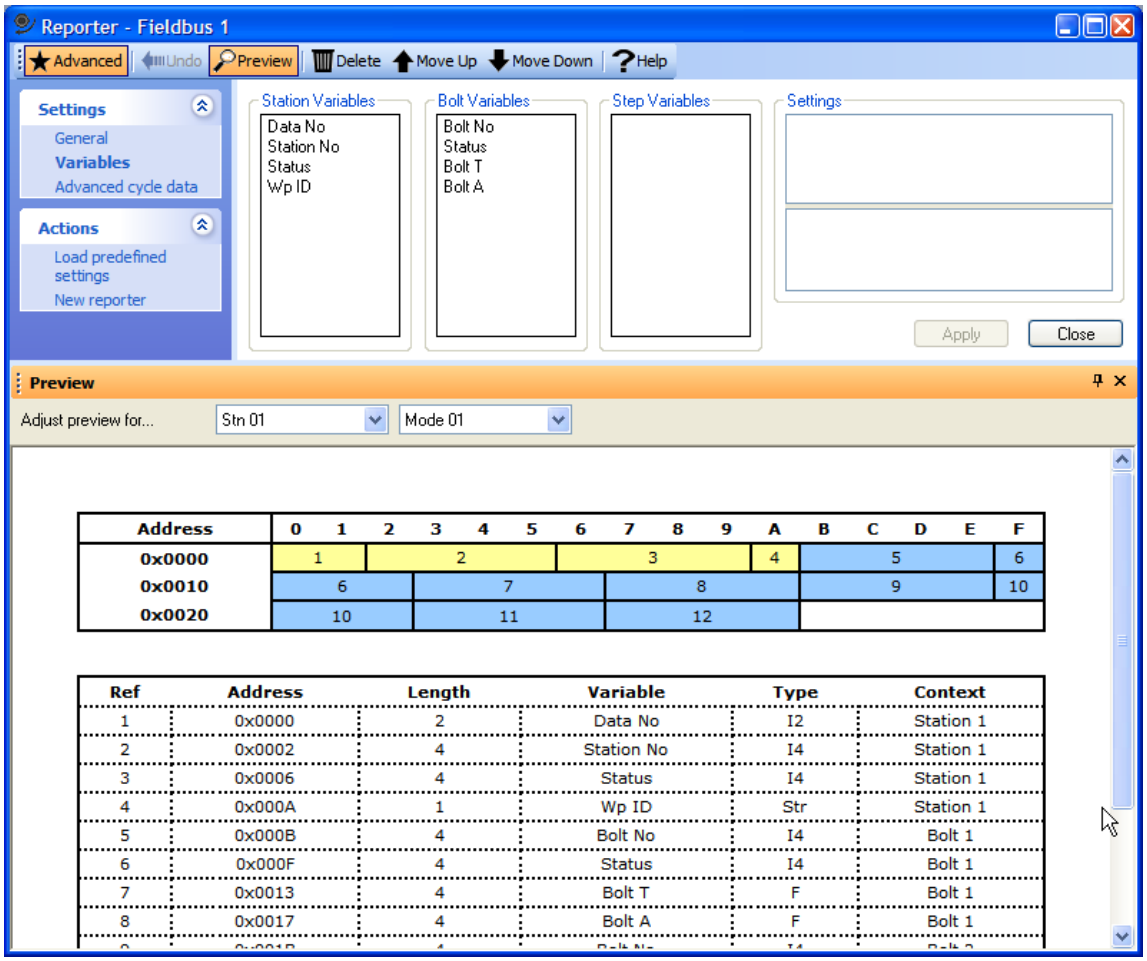
Variable	Width	Dec	Text
Bolt No	8		yes
<New Line>			
Bolt T	7	1	yes
<New Line>			
Bolt A	7		yes
<New Line>			

Gives the following print out for a four-bolt station where only bolt 1, 2, and bolt 4 is executed when printable format is selected:

```
Station: Stn 01  Time: 011030 09:40:58         Status: OK
Bolt No: 1
Bolt No: 2

Bolt No: 4

Bolt T: 15.1
Bolt T: 15.2

Bolt T: 15.4

Bolt A: 431
Bolt A: 432

Bolt A: 434
```

3.7.4.4 Reporter Preview

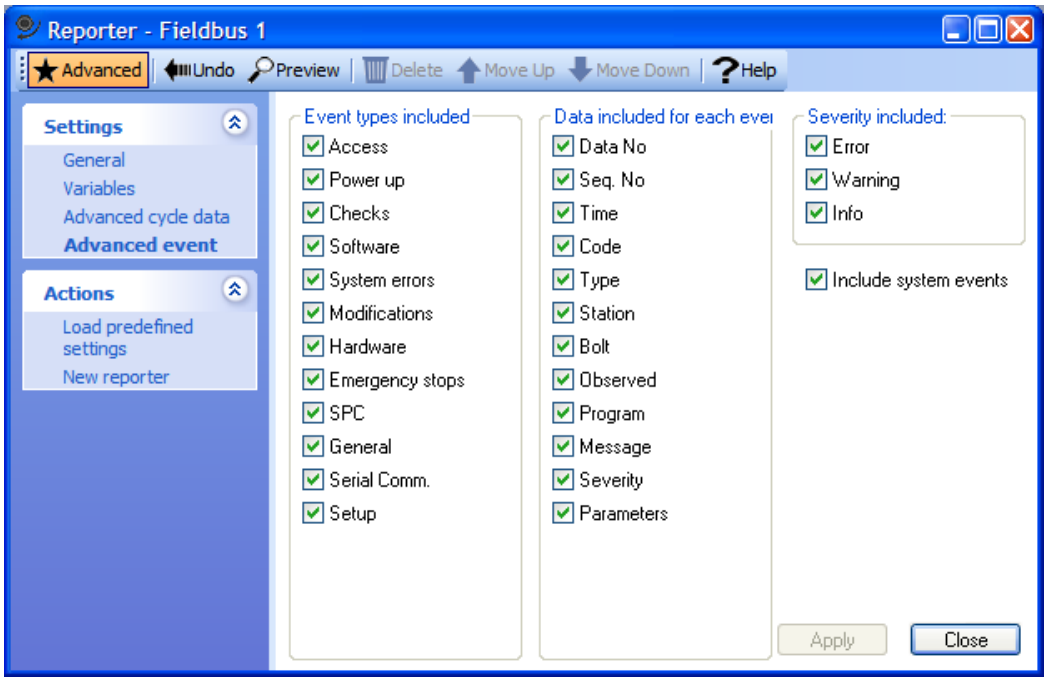
For binary reporters it is possible to preview how the generated data will be formatted.



It is possible to adjust the preview according to a station and mode number. It is possible to hoover with the mouse above the individual fields in the address map to view the actual variable.

3.7.4.5 Advanced Event settings

Using the Advanced Event settings you can configure a filter for which events that should be sent to the device and, for reporters configured to report data in binary format, how they should be formatted.



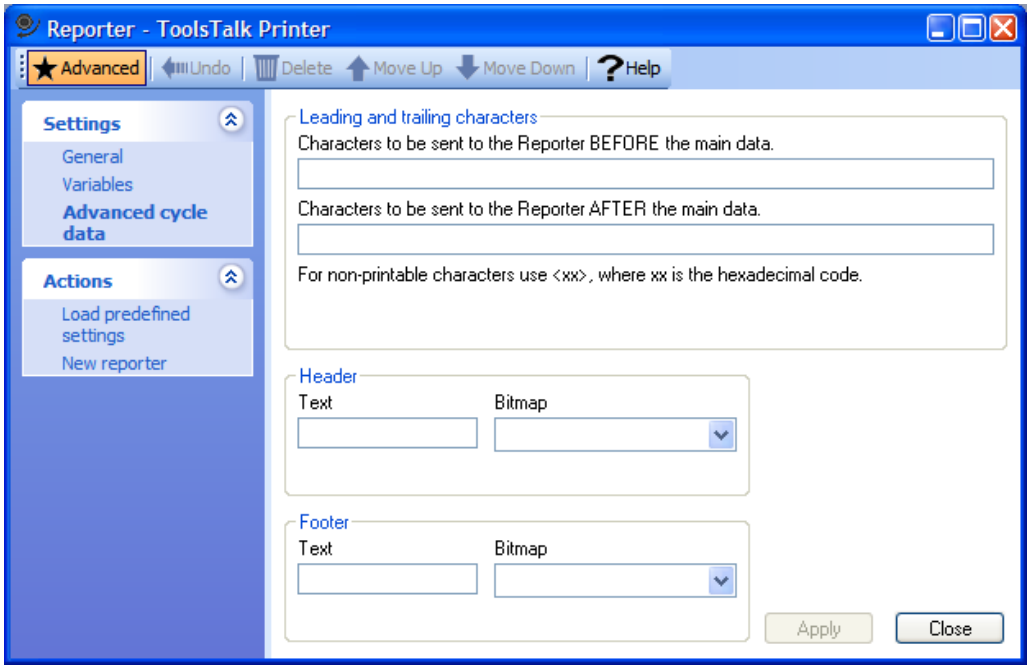
Events can be filtered on their type and their severity. In the **Event types included** frame, check the event types you want to pass to the device controlled by this reporter. In the **Severity included** frame, check the severities you want to pass to the device controlled by this reporter. See View Event Log for a description of event types and severity.

In the **Data included for event when printed binary** frame, check the fields you want to include for each event reported. See Layout of Events for a complete description of the binary format of events.

Include system events controls if events that do not belong to specific station should be reported or not.

3.7.4.6 Advanced CD settings

Given that you have specified a **Header frequency** in the Reporter form you can use this form to add a header and footer to all pages of your cycle data.



Enter the text to be printed at the top of the document in the field **Header on top of page/Text**. For the ToolsTalk Printer you may also use the **Bitmap** combo box to select a bitmap to be displayed as header.

Enter the text to be printed at the bottom of the document in the field **Footer at bottom of page/Text**. For the ToolsTalk Printer you may also use the **Bitmap** combo box to select a bitmap to be displayed as footer.

To be able to include your own bitmap you must first place the bitmap in the directory named **Bmp** located in the directory where you to install ToolsTalk PowerMACS.

Using the fields **Characters to be sent to the Reporter BEFORE the main data** and **Characters to be sent to the Reporter AFTER the main data** you can specify character strings that will surround each cycle data result printed.

Here you may also include non-printable characters. Typing their ASCII code in hexadecimal format enclosed by a "<" and a ">" does this. Example: To frame each cycle data with a start-of-text character (STX), before the data, and an end-of-text character (ETX), after the cycle data then enter:

- "<02>" in **Characters to be sent to the Reporter BEFORE the main data**
- "<03>" in **Characters to be sent to the Reporter AFTER the main data**

Settings for **Fixed pos. layout** are only shown when the **Layout** option is set to **Fixed pos.** The parameter **Report Bolts** controls how many bolts, or actually bolt positions, to use when the bolt result variables are laid out when formatting the bolt variables. Leaving it empty will cause the number of bolts of the station producing the result to be used instead.

3.8 Security

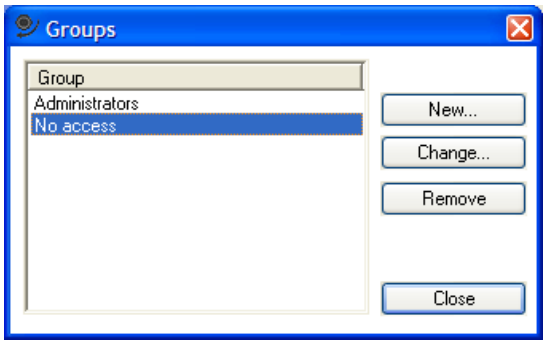
PowerMACS contains a security system that prevents unauthorized access. All personnel must then be registered as users with password and access level.

The access level makes it possible to allow personnel as a group or individuals to have access at these levels:

- No access at all
- Read access
- Read and write access

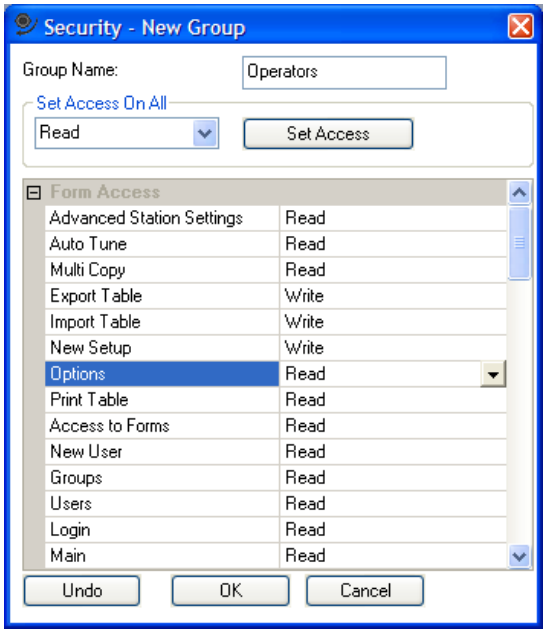
3.8.1 Registering groups

Use menu item **File-Security-Groups** to register a new group, ot to check or change data for an existing group.



By default every setup contains one **Administrators** group that can not be modified or deleted. Users in this group have read and write access to all ToolsTalk PowerMACS forms.

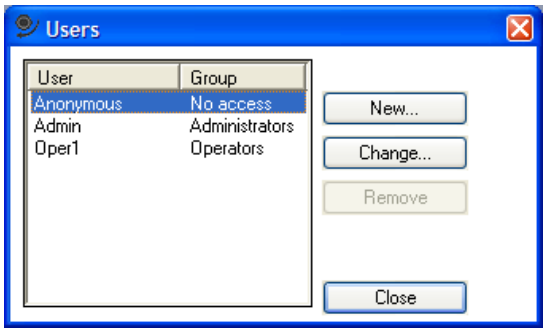
To remove a group select the group in the list and press the **Remove** button. To change group data or create a new group press the **New...** or **Change..** button. The following dialog box is opened



Enter the **Group Name**, use the controls located in the **Set Access On All** frame to set the access of all forms to **No Access, Read** or **Write**. It is then possible to select access level for each ToolsTalk PowerMACS window by selecting the access level in the Form Access list.

3.8.2 Registering users

Use menu item **File-Security-Users** to register a new user, or to check or change data for an existing user.



The list contains current users.

To remove a user, select the name from the list and press **Remove**.

Press **New…** to add a new user.

The following dialog box is shown.

Enter **Name**, **Password**, and select **Group**. Press **OK** to register the user.

To change data for an existing user, select the name from the list and press **Change….** The same dialog box appears as for a new user. Change the data and press **OK**.

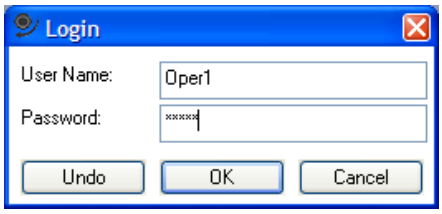
Recommendation: when security is used the group for user “Anonymous” must be changed from Administrators to a group with lower access, preferable the group “No Access” should be used.

3.8.3 Login and Logout

When a new user is about to start to work with the system he should log in. This can be done in either of the following ways:

- Pressing button on the toolbar (if shown)
- Using the menu item **File-Login**

At log in the following dialog box is displayed:



If the name and password is correct the user will be let in. The name of current user is presented on the status bar, down to the right.

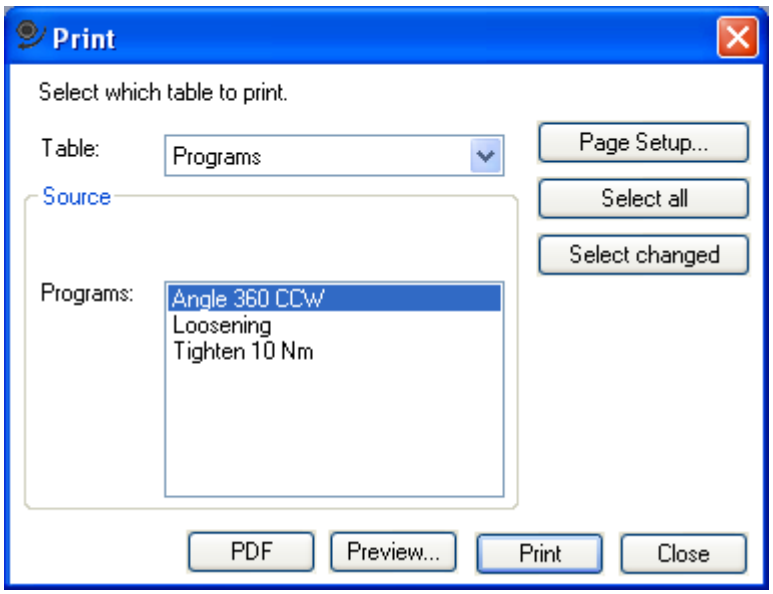
When the user is finished with his work he should log out to prevent other users from making any unwanted changes. This can be done in the same way as for log in.

If no actions are made on the Console for ten minutes the system will make an automatic log out.

If you have made changes to the system and you log out you will automatically be asked to enter a message in the Service Log. You will also be asked to save changes in the setup file.

3.9 Printing

Use the menu item **File-Print…** to print the different tables in the setup.



First select the type of table you want to print in the combo labeled **Table**. This will display all tables of the selected type in the **Source** frame.

Select one table by clicking on it.

If you want to print several consecutive tables, point on the first one, press left mouse button and hold down. Drag the cursor over the table you want to print, and release the mouse button when on the last one.

You can select several non-consecutive tables by clicking on these while holding down the CTRL key on the keyboard.

Press **Select all** to select all tables of the selected type. Press **Select changed** to print all tables that have been changed since last time they were printed.

When more than one table is printed each table is formatted and printed on a separate page.

Press **Page Setup…** to enter a dialog box where you can set up which printer to use, number of copies etc.

When printing programs, you can also get a print preview or save the program as PDF.

Note! You can only print tables on a printer that is accessible from the console computer. Either directly connected or via the network. It is not possible to reach a printer that is connected directly to the tightening controllers in the PowerMACS system.

4 Set Up and Maintenance

4.1 Set Up and Maintenance - Overview

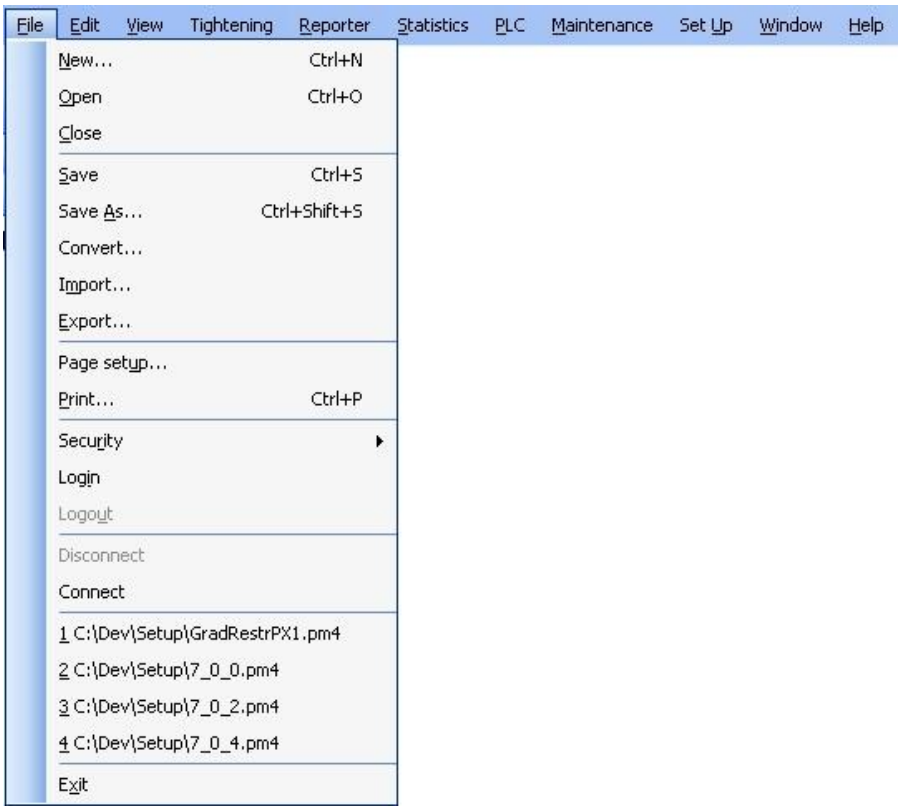
This part handles how to set up and maintain the system.

Note! Be careful when changing data in setups. If you are connected and your system is running, a change will affect the system immediately when you press the **Apply** button in the current form. Even though the change is activated in a controlled fashion between two cycles it might effect your system in a way you did not intend.

4.2 Setups and How to handle them

A *setup* is a group of data that completely describes how a system should work. The PowerMACS system uses the information in the *setup* to know what to do. A *setup* can also be called a *configuration*. A *setup* can be stored on the console computer in a file on a hard disk or on a usb memory stick. It can be downloaded to the system, or uploaded for storing or modification.

Most items on the **File** menu handle setups.



Use the **New** item to create a completely new setup. This is described in chapter: **Set up of a new system**.

With the **Open** item you open an existing setup, to alter or just to check. When you are ready with a setup you can use **Close** to close the setup.

If you have opened a setup and made modifications to it you can save these to disk by selecting **Save**. You can also rename it by using **Save as**. Then the original setup will be kept. This is a also a fast way of creating new setups by starting from a similar one.

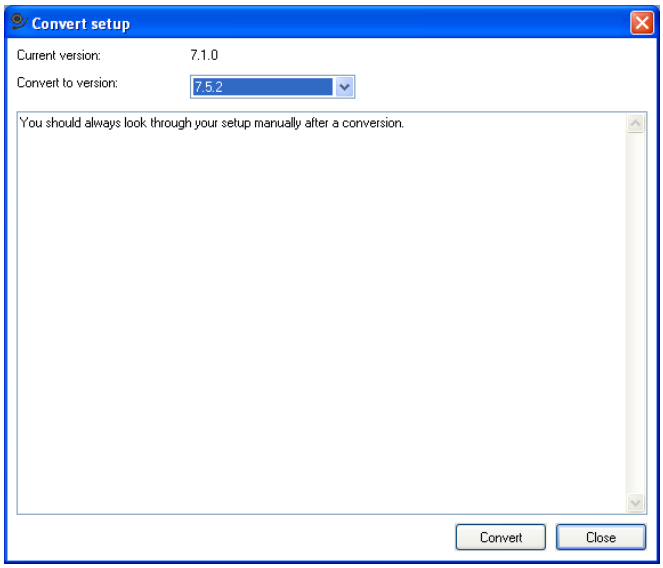
If you have a setup that is too old for the software in the TC, you can **Convert** it to a later version (see section Backwards compatibility).

If you want to copy separate tables from one setup to another you can use the **Export** and **Import** choices. This is described in chapter: Table Export and Import.

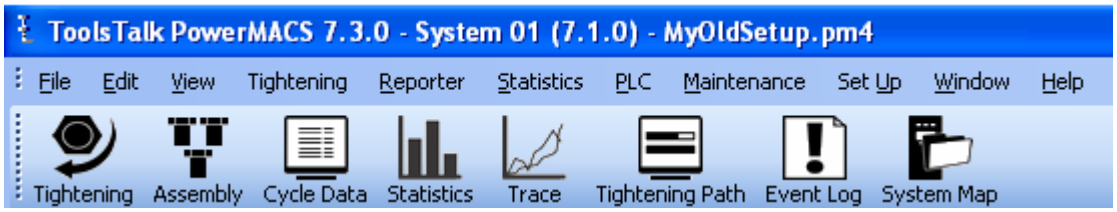
4.2.1 Backwards compatibility

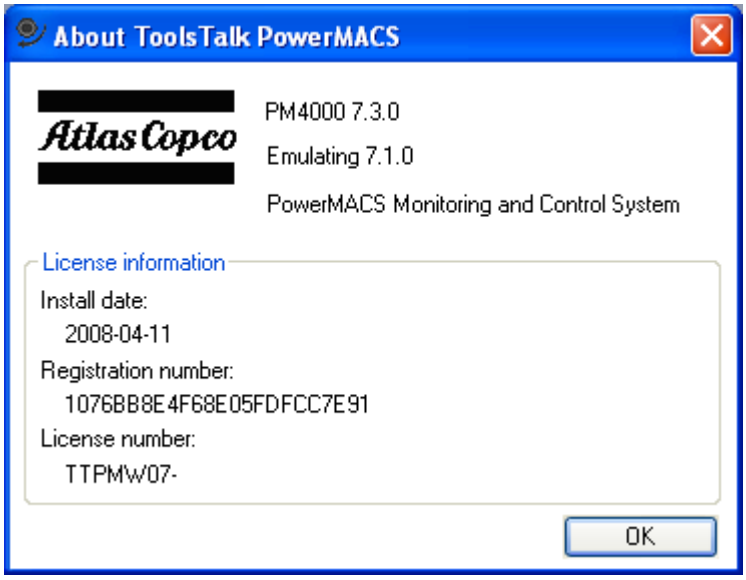
Unless the setup version is identical to the version of the TC software, ToolsTalk will not connect to the TC.

An old setup can be converted to a later version by clicking **Convert** in the File menu. The PLC program will not be converted, but all TC versions can run old PLC programs.



When running an old setup, ToolsTalk will emulate a ToolsTalk with the same version as the setup. Functionality added after that version will be hidden and functionality removed after that version will be shown. The versions of ToolsTalk and the setup are visible on the ToolsTalk title bar and in the **About** form. It looks like the following images when a setup version 7.1.0 is opened with ToolsTalk 7.3.0.





ToolsTalk is only able to emulate versions from 7.1.0 to the ToolsTalk version. Setups older than version 7.1.0 must be converted or an older version of ToolsTalk must be used.

4.2.1.1 PLC conversion

Note: When converting to a version older than 7.5.0, the PLC is not touched at all by the conversion function.

When converting from a pre-7.5.0 setup to version 7.5.0 or newer, the following rules apply for moving PLC variables from the shared area:

Inputs:

1. Variables mapped by any I/O device:
    a. If name starts with “SI_”, variable is left in the shared area
    b. else, variable is moved to the I/O area

   Then, with the remaining variables in the shared area:

2. Variables mapped by fieldbus:
    a. If mapped by any I/O device, variable is left in the shared area
    b. else, variable is moved to the fieldbus area

Outputs:

1. Variables mapped by any I/O device:
    a. If name starts with “SO_”, variable is left in the shared area

b. else, variable is moved to the I/O area

Then, with the remaining variables in the shared area:

2. Variables mapped by fieldbus:

    a. Variable is moved to the fieldbus area

Drivers for the I/O area and the fieldbus areas are added automatically if not already present in the PLC program (the drivers are located in the “I/O Configuration” under “Physical Hardware” in the PLC)

Note 1: It is no longer possible to map an input to both I/O and Fieldbus at the same time. A signal mapped to both I/O and Fieldbus will only be mapped to I/O after the conversion.

Note 2: It is only possible to map an output to both I/O and Fieldbus at the same time if the name starts with SO_. A signal mapped to both I/O and Fieldbus with another name will be moved to the I/O area and is thus no longer possible to map to the Fieldbus.

Note 3: It is no longer possible to remap the individual elements of an array to the Fieldbus, only the whole array can be remapped. If an array is used and some elements have been remapped the whole array will be moved to the Shared_Variables group and removed from the Fieldbus mapping. This means none of the signals in the array will be available after the conversion. If wanted the whole array can be moved back to the fieldbus, but it will NOT be possible to remap the individual elements to get back to the old mapping used before the conversion.

4.3 Set up of a new system

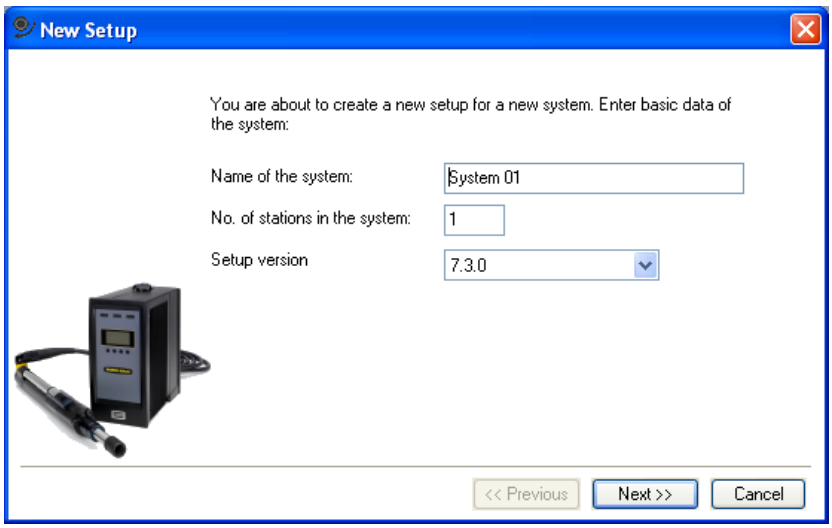
After you have decided what physical hardware you will have in your new system you should make a *setup*. A setup is a file, which contains all information that is needed for the system to execute its task properly.

You can create a new setup *disconnected* on any PC computer. Disconnected means that you are not connected to the actual hardware. When you are ready to run you set up a PC as console computer (CC) on the same network as the hardware and connect. Your PC gets in contact with the hardware and downloads the setup.

Note! You should take as a rule to always create a new setup using the Wizard since it automatically generates a setup that not only is valid but also contains a PLC program.

4.3.1 Creating a new system using the Set Up Wizard

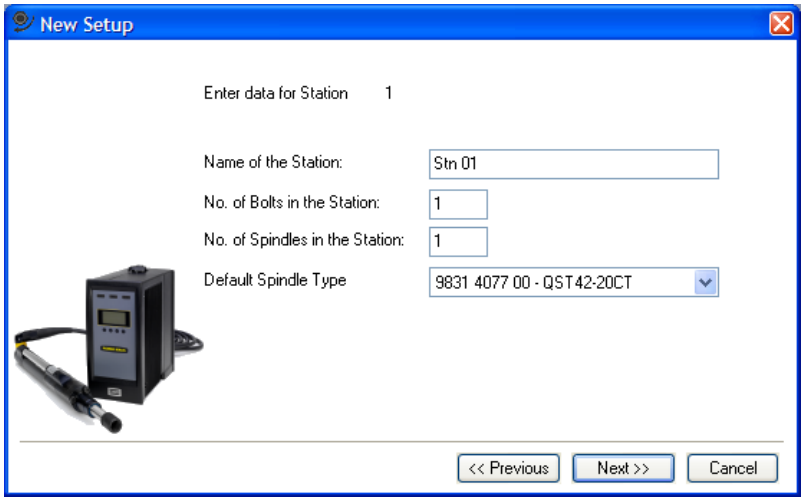
A *wizard* will guide you through the creation of a completely new setup. Select menu item **File-New…**



A wizard is a set of dialog boxes that will lead you through the creation process. If you change your mind you can always go back with use of the **Previous** button.

Enter a name for the system, the number of stations you will have in it, and for which version you want to create it. You should choose the same version as the software in the TC you are going to send the setup to.

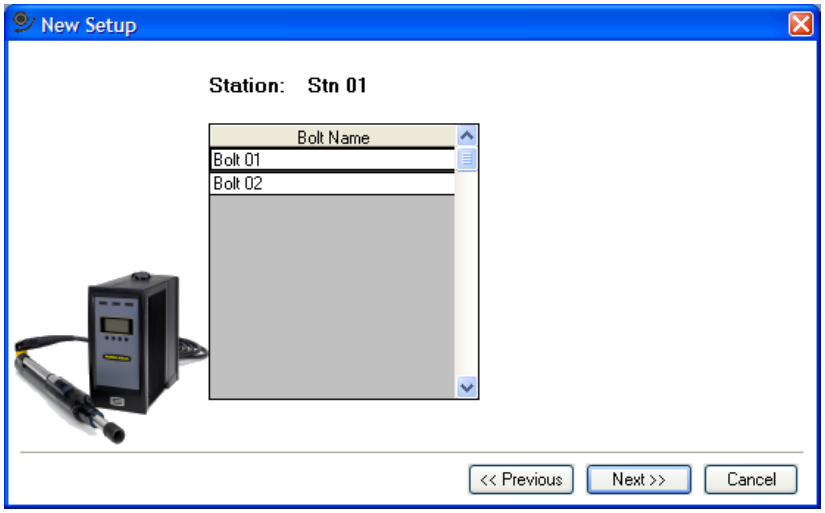
Press **Next >>.** A new dialog box is presented.



Enter the number of bolts and spindles you want to have in the first station using the fields **No. of Bolts in the Station** and **No. of Spindles in the Station**.

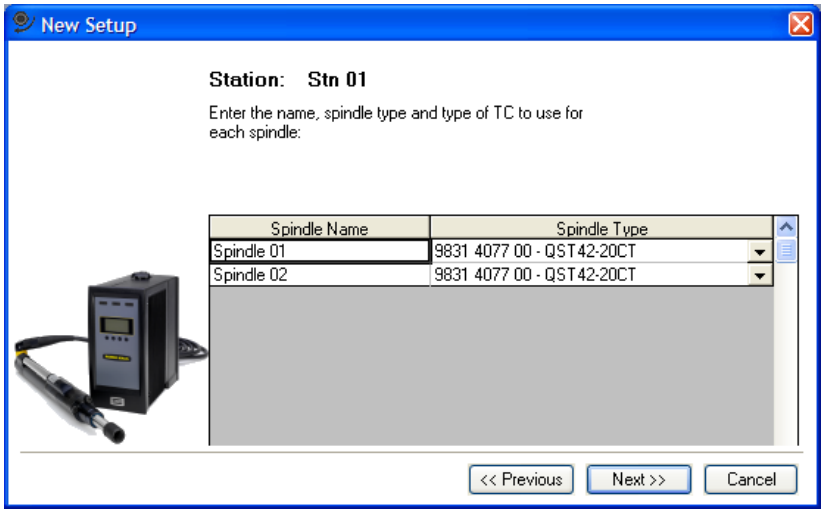
Enter in **Default Spindle type** that you have in your system. If you have more than one type select the alternative you have most of. You can later individually change these values.

Press **Next >>**. This will display the following dialog:



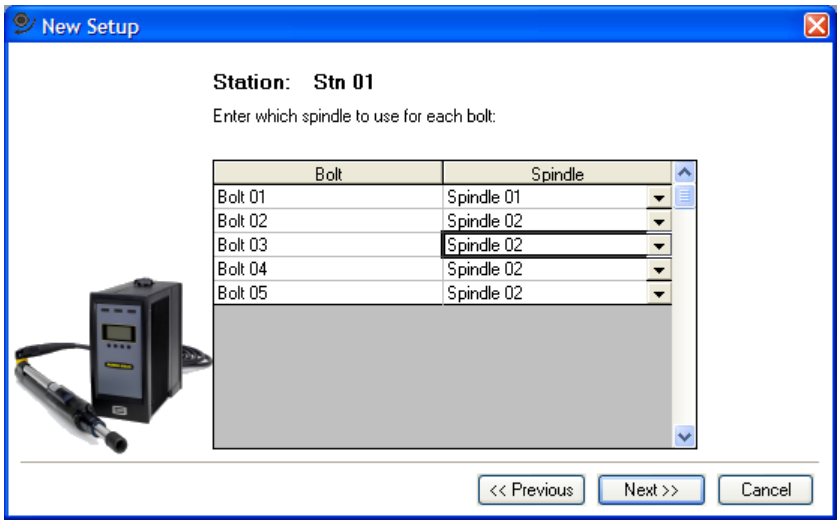
Here you may change the names of the bolts.

Press **Next >>** to be displayed the dialog where you may alter the names given to the spindles:



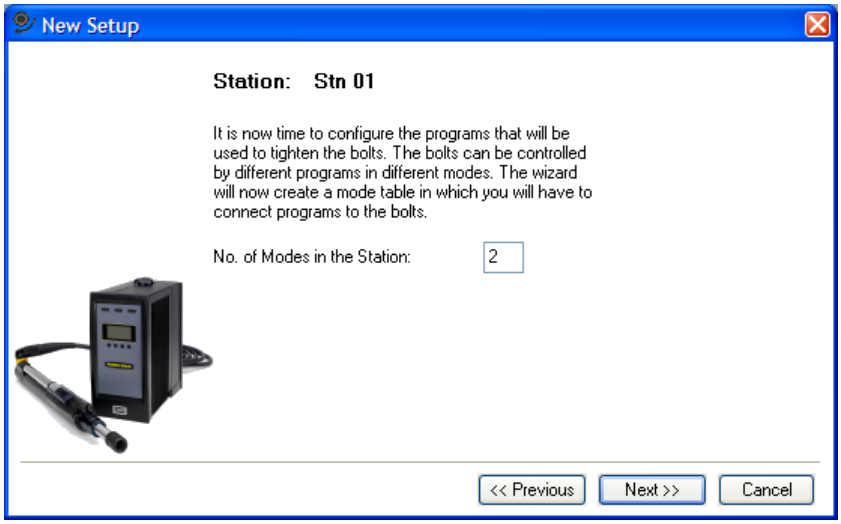
Here you may also change the type of the spindle and the TC corresponding to it.

Press **Next >>**. If you specified fewer spindles than bolts in the second wizard dialog you will be displayed the following dialog:

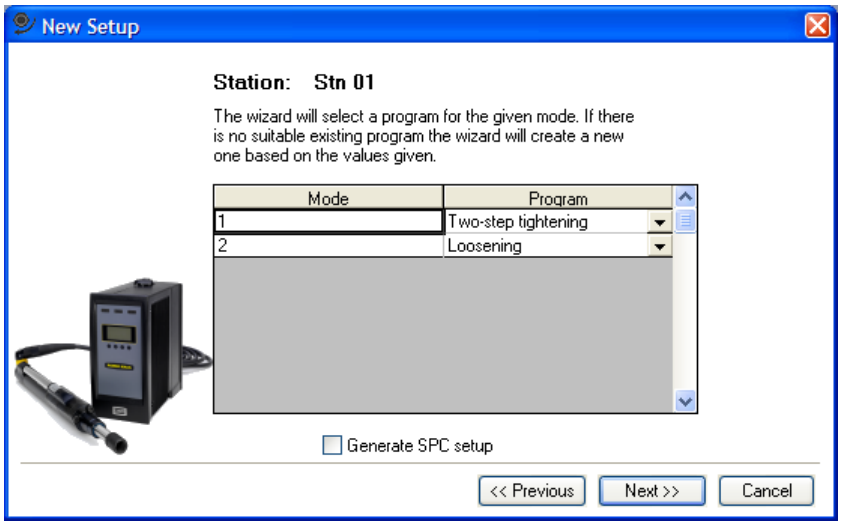


Use the combo boxes in the Spindle column to select the spindle to be used to control each of the bolts.

Press **Next >>** to display the next wizard dialog.



Enter the number of modes you want to have in the Mode table for the station (max is 50) and press **Next >>**. This will present the following dialog:



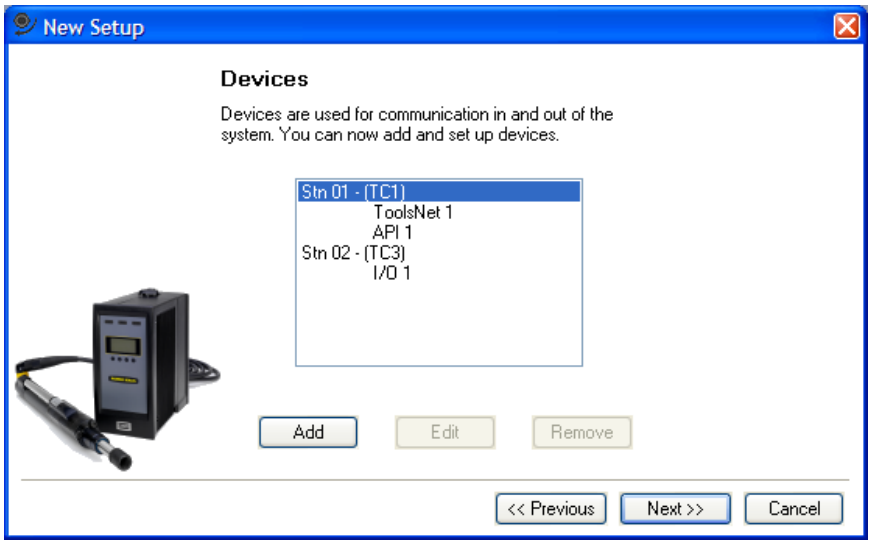
Specify programs for the number of modes selected. You can select predefined programs located in the installation directory for ToolsTalk PowerMACS under the subdirectory **Programs**.

Checking the **Generate SPC Setup** check box will configure the SPC function to calculate SPC data for the variables Bolt T and Bolt A.

When done press **Next >>**. This will cause this dialog to be repeated once for each mode you specified in the previous wizard dialog.

If you specified more than one station in the system the wizard will repeat the sequence above from dialog two for each station.

In the next step you can add and set up *devices* that you will have in your system. Devices are used to get information in and out of the system, like printers, fieldbuses etc.

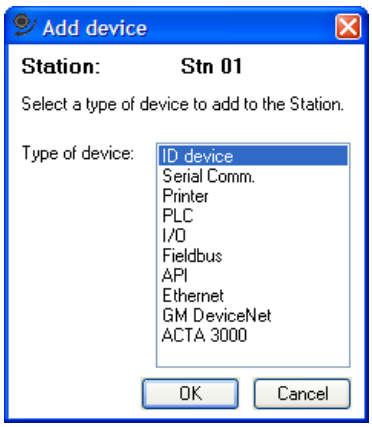


In the listbox all stations and devices are presented.

If you want to change the name or any other parameter for a device, select the device and press **Edit**.

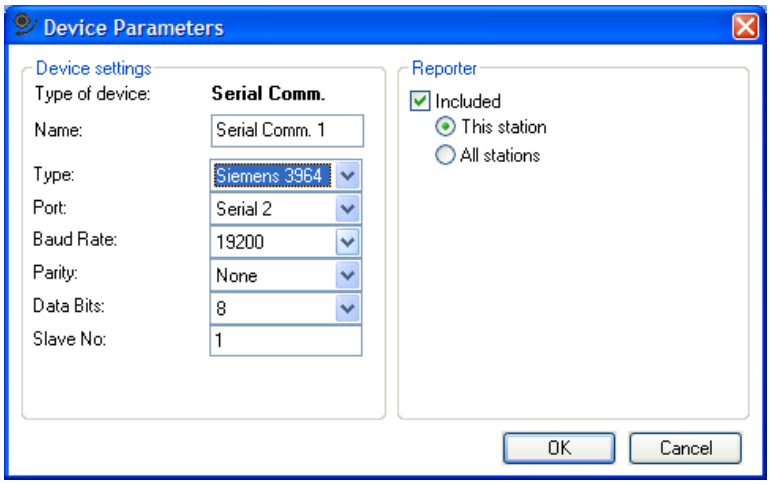
To remove a device, select the device and press **Remove**.

To add a device, select a station and press **Add**. A form is presented where you can select what type of device you want to add.



Devices must be set up to work properly. When you press OK a form is presented where you can set up various parameters.

What data to enter depends on type of device. All parameters come up with default values. If you know what exact values you will have, you can change the values now. You can also wait and change values later, with use of the System Map.

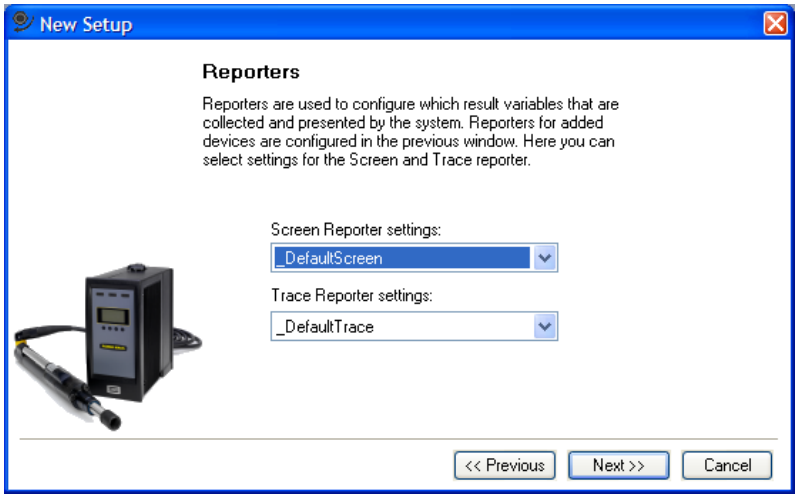


A Reporter is used to format the output of cycle data. Mark the **Include** checkbox if you want a Reporter to be automatically created for this device. Also specify if you want reporter data for just **This station** or for **All stations**.

If predefined reporter settings files exists in the directory for the currently selected reporter type (see **Predefined reporter settings**), you have the option to select predefined settings for the new reporter. If you do not want this, leave the selection blank, which will create the reporter without any settings.

When you have specified all data for all stations included in your system, press **Next >>** to display the next wizard dialog.

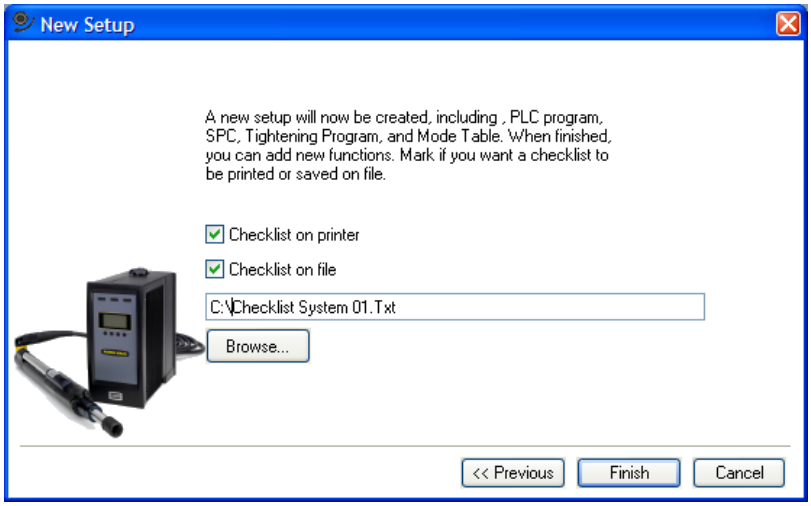
If you have more than one predefined reporter settings file for the Screen or Trace reporter (see **Predefined reporter settings**), the following dialog will be displayed:



Here you can select which predefined reporter settings that will be used for the Screen and Trace reporter. **Note** that it is not possible to leave these selections blank.

Note that this dialog is only displayed if you have placed more than one file in the special directory for the reporter in question. By default, this dialog is **not** displayed.

Press **Next >>** to display the last wizard dialog:



Mark boxes if you want a **Checklist on printer** or **Checklist on file**. In the latter case you can specify a filename. The checklist will contain a list of activities that you should perform after that the initial setup is created.

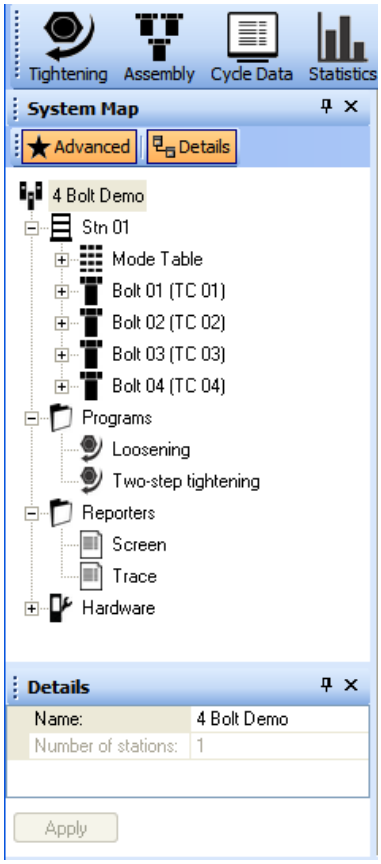
Pressing **Finish** will make the wizard create the new setup based on the information you have specified. The following items will be included in the setup created:

- Spindle and bolt setup’s, one for each item specified.
- Specified programs.
- A mode table for each station specifying the programs you have selected.
- SPC configured for measuring the final torque and angle for each of the bolts.
- A reporter for the ToolsTalk PowerMACS screen and a printer connected to the CC.
- A PLC project containing one POU (Program Organization Unit) and one resource, with one task, for each station defined in the system. All POU's are copies of a predefined POU template.

The setup is complete in the sense that it will be possible to execute. It will therefore be a good starting point for your own application.

4.4 System Map

The System Map by default docked on the left side of ToolsTalk PowerMACS main window is the place to check and alter the system configuration (setup).



You can expand or collapse the tree structure by clicking on an item or clicking in the small squares with + and -.

When you select an item in the tree the **Details** window is updated with relevant data for the selected item, for more complicated items an **Open...** button is displayed that will open an item specific dialog. You drag and resize/place the Details window as you feel most comfortable.

Data in fields with white background can be changed. Press **Apply** to save changes into the setup.

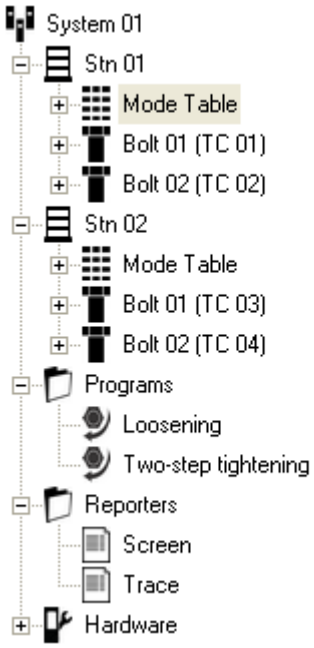
Note! If you are on-line the changes will affect the running system.

The items that make up a system are divided into two parts, the logical structure and the hardware structure. The logical structure is displayed directly under the root node (**System node**), the hardware structure is located under the **Hardware** item.

To add or remove items simply right-click in the system map. However, even though it is possible to add/remove logical items, such as stations and bolts, displayed on the System tab it is recommended that you use the File-New Wizard (see chapter: Creating a new system using the Set Up Wizard) when doing major structural changes to your system.

4.4.1 Logical view

The nodes under System displays the logical items in the setup.



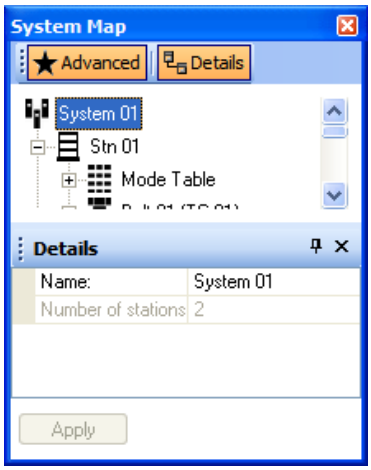
The System is displayed in a tree structure with the System node as the root element. The next level contains all the Stations as well as the Program and Reporters folders.

Each Station contains the Mode table and all Bolts controlled by the Station.

The Programs and Reporters folders contain all programs and reporters that are defined in the setup, these are available to all Stations within the system.

4.4.1.1 System node

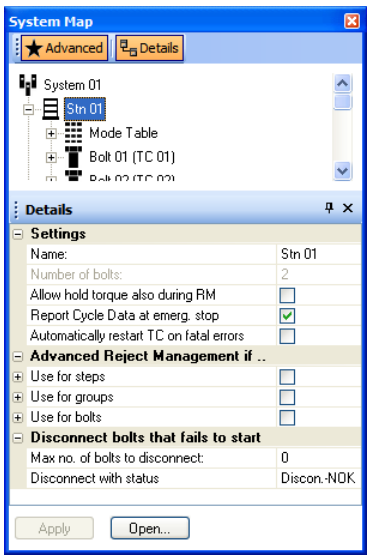
Select the System node by clicking on it using the mouse. This will make all properties of the System to be displayed to in the **Details** window.



Use the **Name** field to change the name of the System. It identifies the system, and setup, and may be up to 20 characters long. The name will be displayed when you scan for System TCs on a PowerMACS network (see Select Target System).

4.4.1.2 Station node

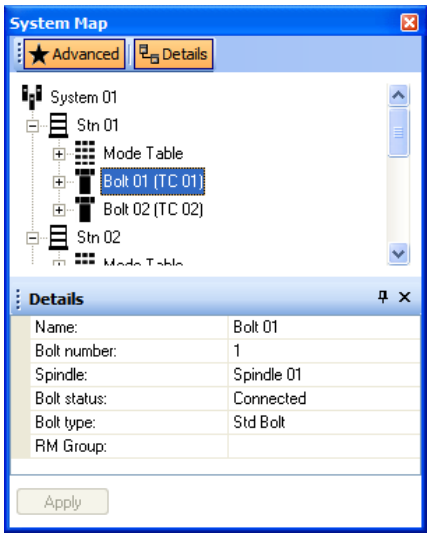
When selecting a Station node many of its properties are in the **Details** window.



See chapter: Station Set Up for a description of the parameters.

4.4.1.3 Bolt node

Select a Bolt node to display the most common properties of the Bolt.



Use the **Name** field to change the name of the Bolt. This name will identify the Bolt in cycle data reports, etc. and may be up to 20 characters long.

The number entered in the **Bolt number** field may be used as a more compact identifier of bolt (in addition to its name). This number may be included in the reports as the variable "Bolt No" (see Bolt level result variables). The number must be unique within the station and within the range [1..9999].

With **Bolt Status** combo box you can control whether or not the bolt is connected, i.e. used by the system, or disconnected. In the latter case you can also chose if it should be reported as OK or NOK.

Bolt type is a free text that identifies the type of the bolt. This information is used for the result variables Total Type, Total Type OK and Total Type NOK, see chapter: Bolt level result variables.

Use the **Spindle** combo box to select the default spindle to use to tighten the bolt. Note that this is only the default selection. A bolt may use different spindles in different modes. See chapter: The Mode Table form for how to configure this.

Specify to which reject management group, or groups, the bolt belongs using the **RM Group** field. Enter comma separated numbers (1, 3, 5), or intervals (1-5). See "Group" in chapter: Glossary for a definition and Step – Rejects for how it is used.

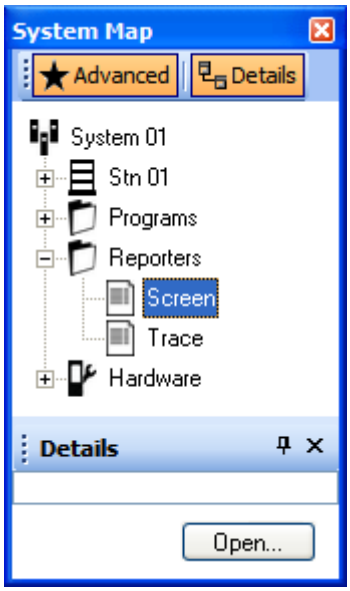
4.4.1.4 Programs node

Clicking on the Programs node will in fold/unfold a list all programs defined in the system.

A program can be opened in The Tightening Program form by double clicking it or right clicking and selecting **Open Program**, to remove a program right click and select **Remove Program**.

4.4.1.5 Reporters node

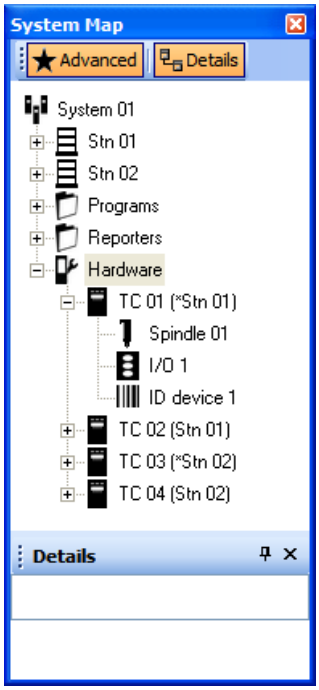
Clicking on the Reporters node will in fold/unfold a list of all reporters defined in the system.



A reporter can be opened in the Reporter form by double clicking a reporter or right clicking and select **Open Reporter**. To delete a reporter right click and select **Remove Reporter**.

4.4.2 Hardware node

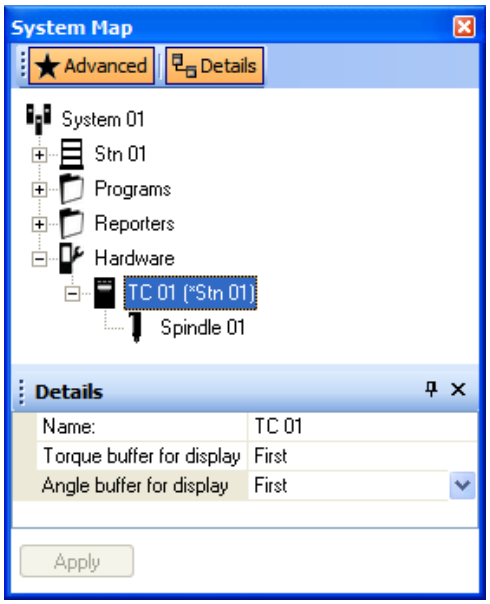
The **Hardware** node of the System Map contains all hardware in the system, this node is the primary resource when configuring or monitoring the hardware configuration.



The tree level below Hardware contains a list of all TCs in the system.

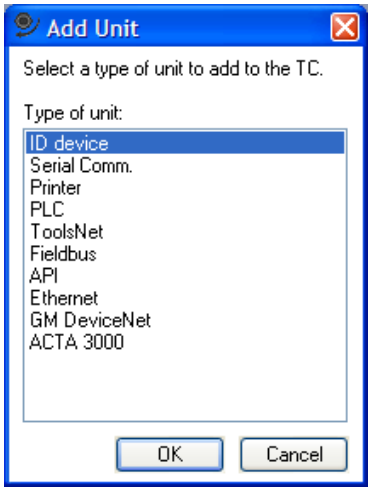
4.4.2.1 TC node

The TC node represent a TC in the system, for each TC it is possible to select which monitoring buffer to use for the TC HMI:s Torque and Angle values.



Under each TC all devices connected to it are displayed as leafs. Normally all TCs have a Spindle device. To the right of each TC it sais which station the TC is running under and whether it is a Station TC (indicated by a star as prefix to the station name).

In addition to the spindle device you might add devices as you need them. The PowerMACS system supports numerous types of devices. To add a device to TC first select the TC to which you want to add a device and then right click and select **Add Device**. This will bring up the Add Unit form in which you select the type of device to add.



Most devices that should handle result data require a Reporter to function properly. The Reporter controls what data that is reported over the device, and how it is formatted.

For devices that **must** have a Reporter, a Reporter is added automatically. For devices that **can** have a Reporter you will be asked if one should be added. In both cases, if a predefined Reporter exists for the device type added, you will be offered to choose one of them (see Predefined reporter settings for how to add your own). If a reporter is added it must also be configured. See chapter: Edit reporter for how to do this.

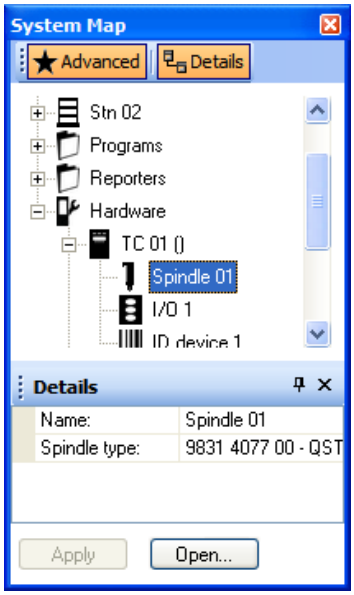
When ToolsTalk PowerMACS is online the TC popup menu contains a **Manage Software** alternative. When this alternative is selected the **Software Versions** form is opened.

This form lists the running software versions for the **Servo** and **Spindle** and also the software versions stored in the TC:s flash memory. The servo and spindle software is not automaticly upgraded – you should always verify with the release notes if it is necessary to upgrade the servo or spindle software.

Note! Upgrading the spindle software might require re-programming of spindle parameters and motortuning.

4.4.2.2 Spindle node

Select a Spindle node to display the most common properties of the Spindle.



Use the **Name** field to change the name of the Spindle. This name will identify the Spindle in cycle data reports, etc. and may be up to 20 characters long.

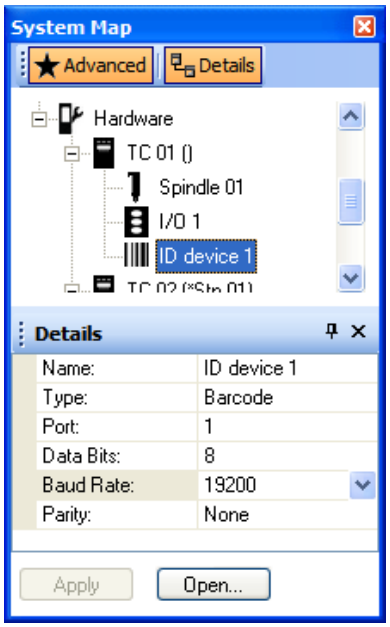
Use the **Spindle type** combo box to change the type of the used spindle.

Note! Changing spindle type will reset all spindle parameters to their default values. Any changes done compared to the previous types default values are lost.

Press the **Open…** button or double click on the spindle node to access the Spindle Set Up form that enables full access to all the spindle parameters.

4.4.2.3 Other device nodes

Selecting any node of any other device type will display the parameters of the selected device type.



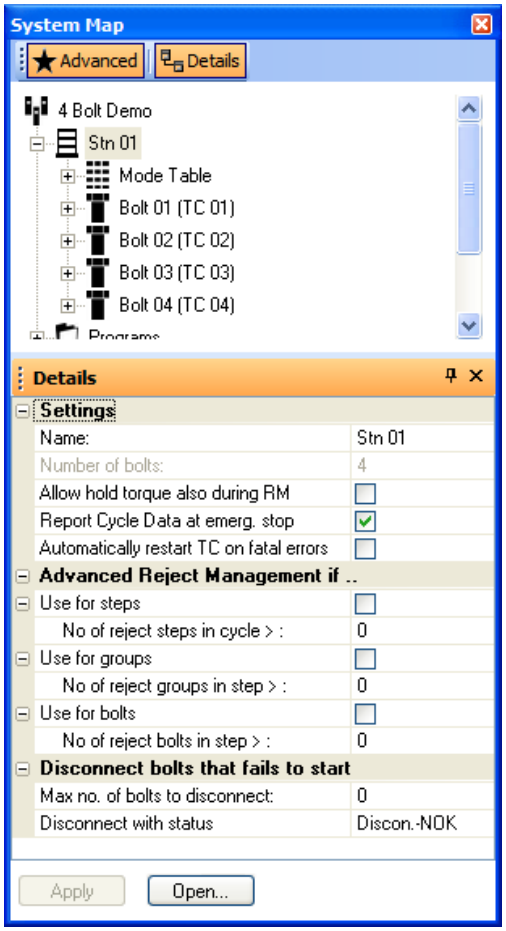
For most devices all parameters are displayed in the **Details** window.

Devices that have many parameters make use of a specific form to access these. For such devices an **Open…** button is displayed. Press this button to open the device specific form.

The basics of these devices are described in the Peripheral Devices chapter.

4.5 Station Set Up

Select a Station node object on the System tab and make sure the **Details** window is visible to display the parameters available for a Station.



The name entered in the **Name** field defines the name of the Station. It will identify the Station in cycle data reports, etc. The name may be up to 20 characters long.

The **Advanced Reject Management if…** settings are described in the functional description for Tightening, see chapter: Advanced RM Actions.

Normally the **Hold Torque** step stop condition (see Tightening Program chapter: Ramps & Other – Other) is automatically disabled for all bolts when a step fail on any of the bolts involved. This is done to avoid overheating the motors. If it is absolutely necessary the torque may be maintained also during the reject management phase. Check the **Allow hold torque also during RM** checkbox to do this.

Note! Allowing hold torque during RM dramatically increases the risk of burning the motors!

Report Cycle Data at emerg. stop controls whether or not cycle data is reported for a cycle interrupted by an emergency stop (caused by setting the PLC variables EMERGSTOP or MACHINESTOP, or if the Station loses communication with one or more of its TCs). Uncheck it to automatically drop all result data for such cycles.

With the **Automatically restart TC on fatal errors** parameter you can control whether or not the TCs used by this Station should reset themselves if they run into an unrecoverable software error. If the TC running the Station controller is automatically restarted it will **always** set its own status, and the status of all its bolts, to NOK to make sure that a NOK cycle is not reported as OK by accident. This is done also if the Station was not running a cycle when it restarted. All TCs that is restarted due to a fatal error will generate an event in the event log when they do so. More detailed information on the error causing the restart can be retrieved using the TC Crash Log.

With the **Disconnect bolts that fails to start** parameters the Station can be configured to automatically disconnect bolts that cannot be started either due to a communication error (Ethernet) or a servo error.

Set **Max no. of bolts to disconnect** to the maximum number of bolts that may be disconnected automatically due to communication or servo errors. Possible values are between from zero up to the number of TCs in the Station minus one.

The radio buttons **Select Disconnect with status OK** and **Disconnect with status NOK** controls with which status the bolts are reported if they are disconnected.

Note! It is not recommended to use the Select Disconnect with status OK choice since it will cause non-tightened bolts to be reported as OK.

As long as the total number of automatically disconnected bolts is less than **Max no. of bolts to disconnect** the event "Failed to reach TC. Bolt is disconnected (OK/NOK)" is generated for each TC that cannot be contacted. When the maximum number of disconnects has been reached the event "Station has no contact with spindle X" is generated for each additional failing bolt.

If at least one, and less then or equal the maximum number of allowed, bolt(s) have been disconnected the event "Station starts the cycle even though X spindles could not be reached" is generated.

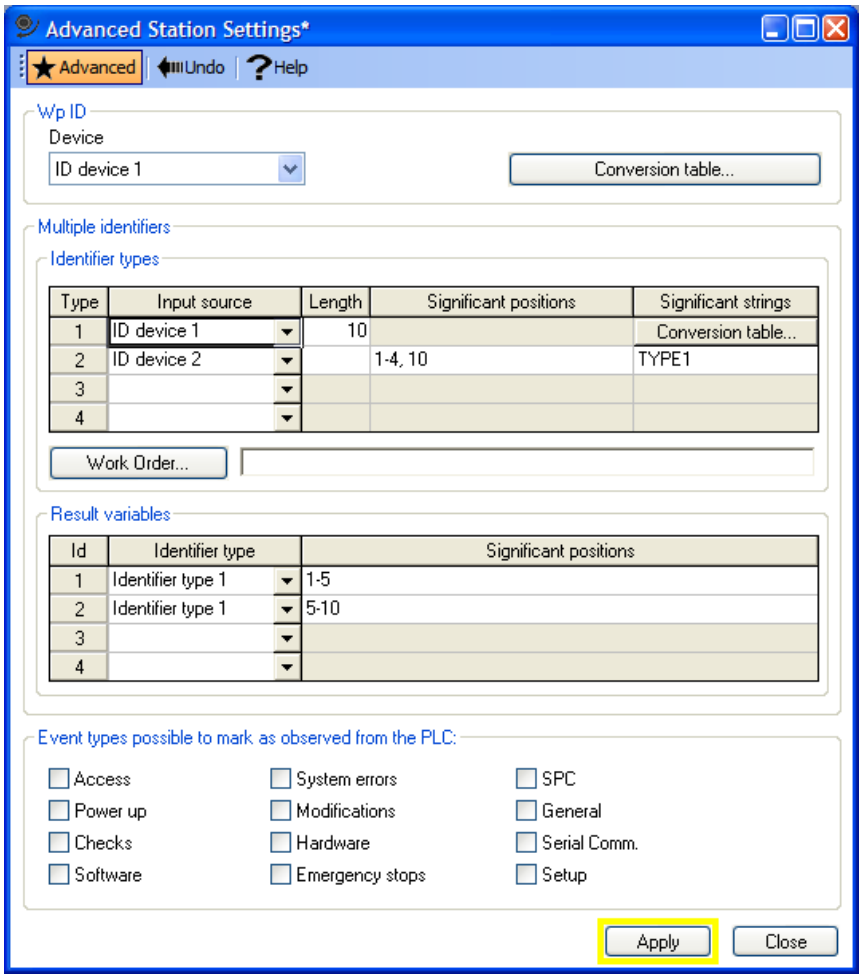
For each bolt that is disconnected due to servo failure during initialization the event "Failed to init servo. Bolt is disconnected (OK/NOK)" is generated.

Press the **Open…** button to open the Advanced Station Settings form for additional settings, alternatively you can right click on a station and select Advanced Station Settings from the popup menu.

4.5.1 Advanced Station Settings

The Advanced Station Settings form is invoked by pressing the **Open...** button in the System Map when the selected node is a Station, alternatively it is possible to right click on a station and select Advanced Station Settings from the popup menu.

From this form you can set up which ID device to use as the Stations standard Work piece identifier (Wp ID), configure the Multiple identifiers function, and control its PLC Event handling.



4.5.1.1 Work piece identifier

The Work piece identifier (Wp ID) function makes it possible add an identifier string read from an ID device, for example a barcode scanner, to the result data as the result variable "Wp ID" (see Station level result variables).

Use the **Device** combo box to select the ID device this station should use as source for the Wp ID function. The combo box lists only the devices that already exist in the system. See chapter: Add a device for how to add a new one.

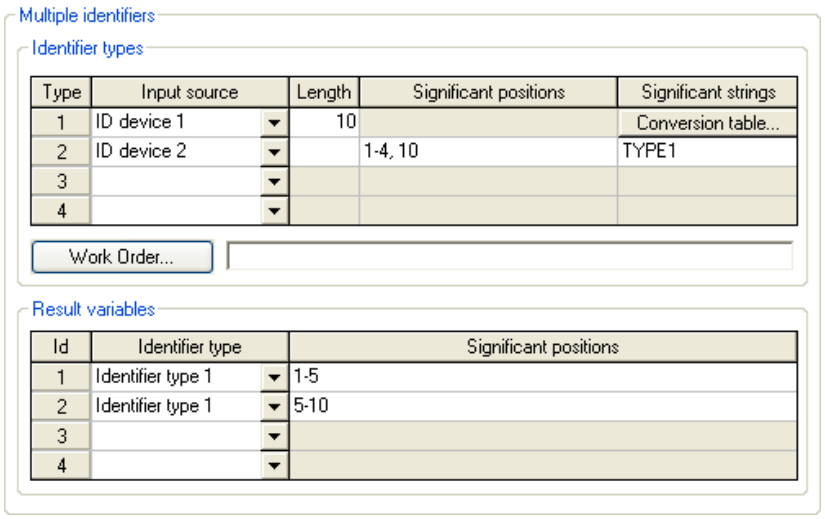
The identifier string read form the selected device is also available in the PowerMACS PLC as the input CURRIDSTRING.

By adding a **Conversion table...** you may have the identifier string automatically translated to a numerical value, or code. This code is then available to the PLC as the input IDCODE and can, for example, be used for selection of which mode to run. See chapter: ID device variables for a description of the respective PLC variables.

4.5.1.2 Multiple identifiers

Using the Multiple identifiers function PowerMACS can handle up to four different types of identifier strings, supplied from the same or different ID devices.

The Multiple identifiers function does not only add received strings to the result data, it can also be used to differentiate strings supplied from one source depending on their data and/or control in which order they are accepted.



The Multiple identifier function can be monitored and controlled from the PowerMACS PLC using its Multiple identifier variables.

Multiple identifiers are defined in two steps using **Identifier types** and **Result variables**.

Identifier types

An **Identifier type** is a string supplied by an identifier **Source**, that is, a device that supplies the identifier data, and fulfills an optional *match condition*.

Any of the systems ID devices but the one configured for the Work piece identifier function can be used as **Source**. In addition to these the four PLC outputs PLCx_IDTYPE_STR (see chapter: Multiple identifier variables) can be used. These make it possible to supply identification data from other device types, for example fieldbus, API, etc.

The *match condition* is specified using the **Length** parameter and the combination of the **Significant position** and **Significant string** parameters. For a string to match it must fulfill all the following rules:

- If a **Length** is specified then the incoming string must have exactly **Length** characters to pass. Leave empty to accept strings of any length.
- If **Significant position** and **Significant string** are non-empty then the specified character position are first picked out from the incoming string. These must then match the regular expression specified by the significant string (see chapter: Conversion table for a description of regular expressions). Leave empty to accept all strings regardless of their content.

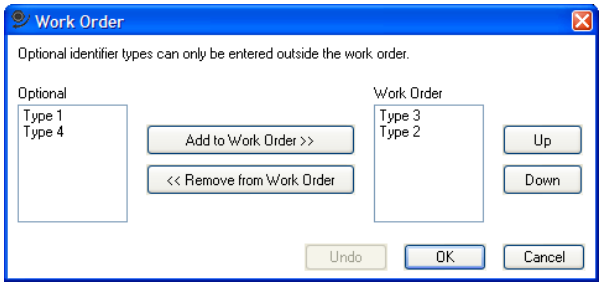
For Identifier Type 1 up to twenty significant strings can be specified using a Conversion table. Aside from that each of these strings are tested, one by one until a match is found, the same matching rules applies. With the Conversion table it is also possible to have the matching identifier string automatically converted to a numerical value available in the PLC as input IDTYPE_1_CODE (see PLC chapter: Multiple identifier variables).

If the same source is used for more than one Identifier type and the respective *match conditions* does not guarantee that an incoming string matches only one of the types then a rule must be set up for in which order the Types are assigned. Press the **Work Order...** button to open a form where this can be done.

The Identifier type strings are not directly available in the result data produced by a cycle. They are however available in the PLC for evaluation.

Work Order

A Work Order is useful when PowerMACS should receive the Identifiers in a given order and/or to resolve in which order the received strings are assigned to the Types if they are ambiguously defined.



The Identifier types included in the Work Order must be received in the order that they are listed to be accepted.

If there is at least one Type in the Work Order list the Multiple identifier function will always try to match this Type before testing any type outside the Work Order.

The Work Order is said to be *passive* until its first Type is accepted. When this takes place the Work Order becomes *active*.

While the Work Order is *active* then only an input string that matches the next, not already accepted, Type in the Work Order list is accepted. All other strings are neglected even if they would have matched another Identifier type.

When the last Type of the Work Order list is accepted the Work Order sequence is completed. This will put the Work Order in passive mode, waiting for the first Type again.

Types that are not included in the Work Order may be received, and are accepted, in any order as long as the Work Order is *passive*.

Types that are not uniquely defined, meaning that they use the same source and have *match conditions* that does not ensure that only one of them match a given input string, should be included in the Work Order. The Work Order then controls in which order received input strings are assigned to the respective Type.

Note Using ambiguous Types makes the operator, or overriding control system, responsible for supplying the identifiers in the expected order.

The values of the Identifier Types included in the Work Order are not available to the Station as long as the Work Order is *active*. On completion, all Identifier Types of the Work Order are sent to the Station and its PLC.

Should the Work Order become active again after completion (by receiving the first Type in the list) then all Identifier Types of the Work Order are reset.

The values of the Identifier Types which are not included in the Work Order are available to the Station and PLC immediately when accepted.

The Multiple identifier function is reset, meaning that the values of all Identifier Types are cleared, when a cycle is started from the PowerMACS PLC. That is, accepted identifiers can only be used once. To include them in a following cycle their respective source must be initiated to generate the string again.

Result variables

To include the data of an **Identifier Type** string in the result data it must be mapped to one of the four **Result variables**. Each of these result variables is represented in the cycle data by one of the for Station level result variables "ID Res 1" to "ID Res 4" (all 40 character strings):

This is done by selecting the wanted Identifier Type as source for a particular result variable in the **Result variables** frame. One Identifier Type can be used as source for more than one result variable.

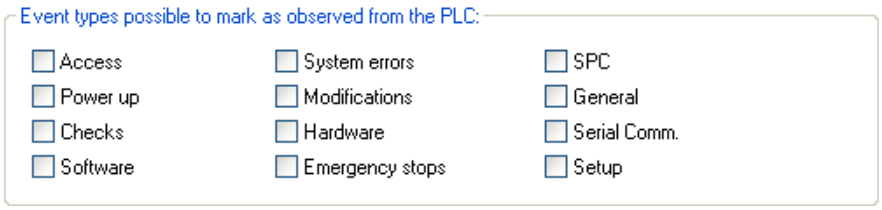
Use the optional **Significant position** filter to select which character positions of the Identifier Type string to transfer to the result variable.

The values of the result variables are sampled when the cycle is started from the PLC.

4.5.1.3 PLC Event handling

The Advanced Station Settings form also enables you to configure which type of Events that should be possible to mark as observed from the PowerMACS PLC using the outputs named ACKxxxEVENT.

Only the types checked in the **Event types possible to mark as observed from the PLC** frame will be possible to acknowledge from the PLC.

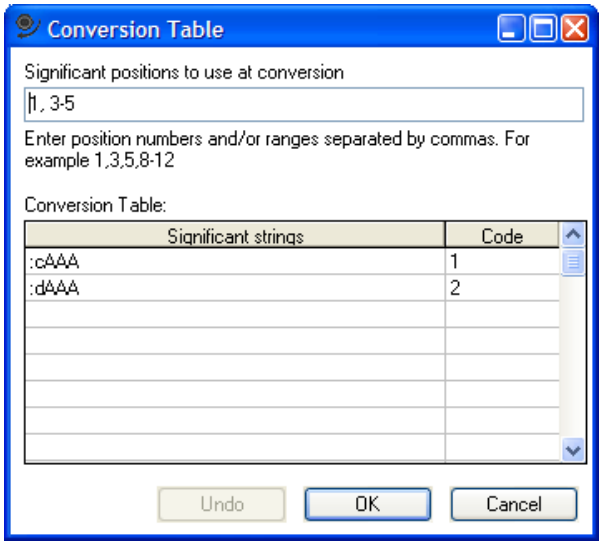


From inside the PLC it is possible to see for which event types there exist unobserved events in the Event Log. This information is available via the inputs xxxEVENT (SPCEVENT, ACCESSEVENT, etc.).

See View Event Log for a description of the different Event types and the PLC chapter: Station variables for a description of the PLC inputs and outputs.

4.5.1.4 Conversion table

The Conversion table consists of a character mask name **Significant positions** and a list of up to 200 pairs of regular expressions, or **Significant strings**, and **Code** values.



The conversion table function starts by running the input string through the **Significant position** filter resulting in a string consisting only of the characters at the specified positions of the input string.

This string is then tested against the **Significant strings** one by one, starting from the top, until a match is found.

If a match is found, and a **Code** is specified for the matching string, its value is returned. If there is no match, or the **Code** column is empty, the result is set to NOT_DEFINED (-32768).

The resulting Code is available in the PLC as input IDCODE if the table is a part of the Work piece identifier configuration, or as the input IDTYPE_1_CODE if it is a part of the Multiple identifiers setup. See PLC chapters: Station variables and Multiple identifier variables for a description of the respective input.

The **Significant strings** may consist of all characters having an ASCII code between 31 and 126. The exact number of entries (N) that can be entered depends on the length (L) of the **Significant strings** according the following formula: N = MIN(200, 4500 / (L + 3)). Example: If all strings are 27 characters long then 150 rows can be entered.

Significant string syntax

```
<Significant string> ::= <RegularExpr> <OptRangeCheck>

<RegularExpr>        ::= <Expression> <OptExpression>

<OptExpression>      ::= <Expression> <OptExpression> | NOTHING

<Expression>         ::= See table below.

<OptRangeCheck>      ::= ';' <TagRef> <RelOp> <IntegerValue> <OptRangeCheck> |  NOTHING

<TagRef>             ::= '\'N  where N is a number from 1 to 9. Represents the value of
                         the Nth "tagged expression" found in <RegularExpr> from the left.
                         See the below table for a definition of a "tagged expression".

<RelOp>              ::= '<' | '>' | '=' | '<=' | '>='

<IntegerValue>       ::= A sequence of decimal digits representing an integer number in
                         the range [0..999 999 999]
```

Expression	Syntax	Description
Any one character	`?`	Matches any one character.
Zero or more characters	`*`	Matches zero or more occurrences of any alphanumerical character.
One alphabetic character	`:c`	Matches one alphabetic character [a-zA-Z].
One decimal digit	`:d`	Matches one decimal digit [0-9].
One hexadecimal digit	`:h`	Matches one hexadecimal digit [0-9A-F].
Tagged expression	`{}`	Tags the text matched by the enclosed expression. The enclosed expression must either be a sequence of decimal digits or a sequence of hexadecimal digits. The value of the enclosed expression can be referred to in a subsequent range check (see `<OptRangeCheck>`). Example: {ddd} or {hhhh}
Escape	`\`	Matches the character following the backslash (\). This allows you to find characters used in the regular expression notation, such as { and :.

Set of characters	[]	Matches any one of the characters enclosed within the []. Example: [abc] matches any single occurrence of the characters 'a' 'b' or 'c'.
Range check start	;	Marks the start of a range check (see <OptRangeCheck>)

Example

A Conversion table with Significant positions set to 1-10, 24 and the following Significant string / Code pairs:

Significant string	Code
L?????????1	11
L?????????2	12
S?????????1	21
S?????????2	22
[AB]{ddd}?{ddd}???; \1 < 300; \2 < 700	1
[AB]{ddd}?{ddd}???; \1 >= 300; \2 >= 700	2
E*	100

will give the following result:

Input string tested	Result
L123456789X00000X11111X2	12
L1234567892	-32768
S123456789X00000X1111111	21
E299X666AX12345678901234	100
A200Y666AX12345678901234	-32768
A500X888AX1	2
B200Z666AX1	1

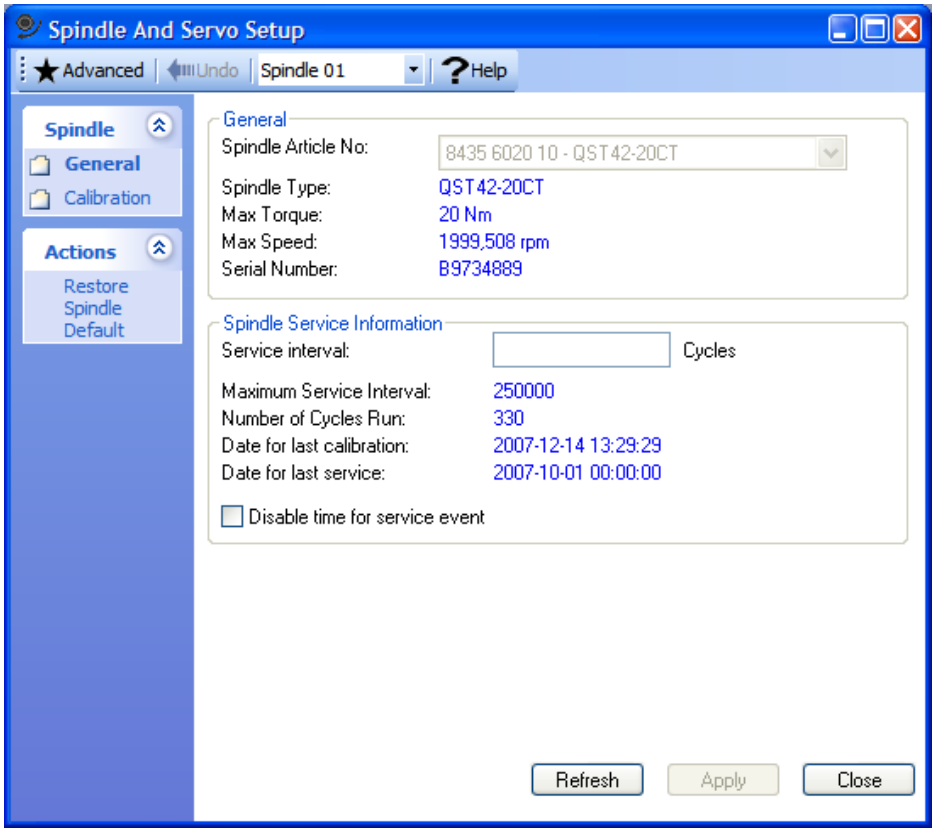
4.6 Spindle Set Up

There are many parameters that control the running of a spindle. During set up these parameters are given default values. In some cases this is not enough. Therefore it is possible to change all parameters with the Spindle Set Up form invoked by selecting the **Spindle** item in the **Set Up** menu or by double clicking the spindle in the System Map.

The controls on the Spindle Set Up form is divided in several pages. Each of these is described in the following subsections.

4.6.1 General page

The General page holds the most basic spindle parameters.



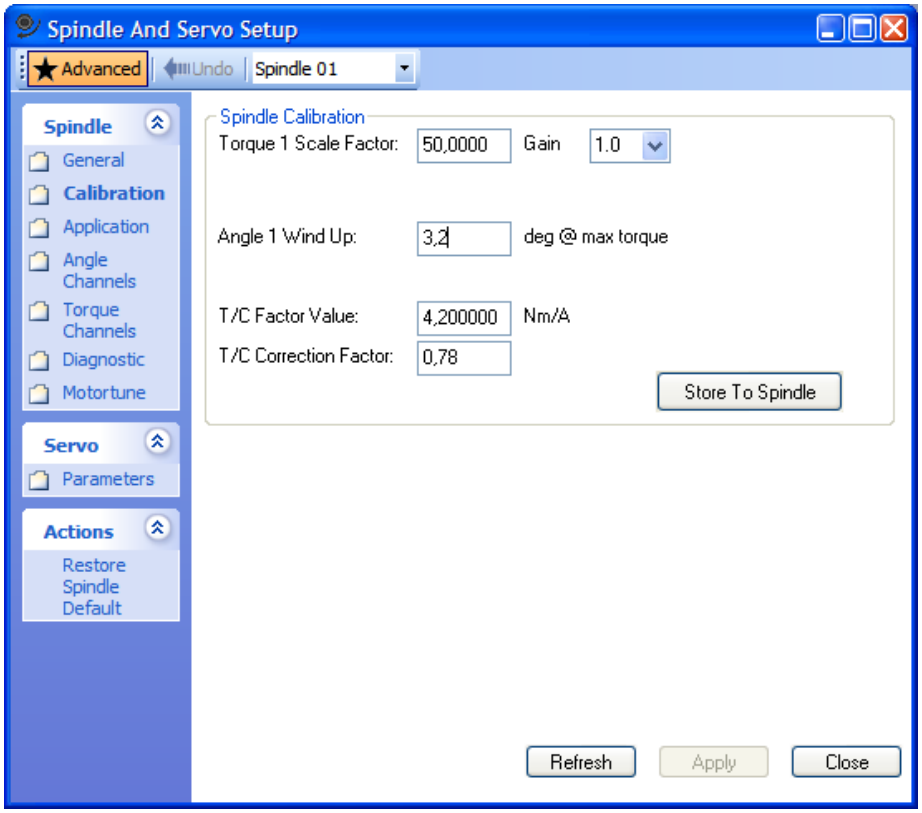
Use the **Spindle Article No** combo box to select the type of spindle to use. It will list the type name of all spindle type files (extension `.ttt`) located in the directory `SpindleType` (a sub directory to the directory where ToolsTalk PowerMACS is installed). It is necessary to have the same article number specified in the setup as the actual hardware used.

To use a conversion box for QMR/QMX spindles, see appendix Configuration of the Conversion Box.

The parameter **Service interval** is used to define the service interval, in number of cycles. To disable the time for service event check the **Disable time for service event** checkbox.

4.6.2 Calibration page

The Calibration page is used to calibrate a spindle. All parameters on this page are stored to the physical spindle. Note that only the available channels are visible.



The parameter **Torque 1 Scale Factor** and **Gain** controls the measuring on Torque Channel 1. Normally a spindle with a 50Nm transducer should have a scalefactor of 50 at Gain 1.0. If the spindle has a second torque channel equipped it is possible to calibrate that channel aswell.

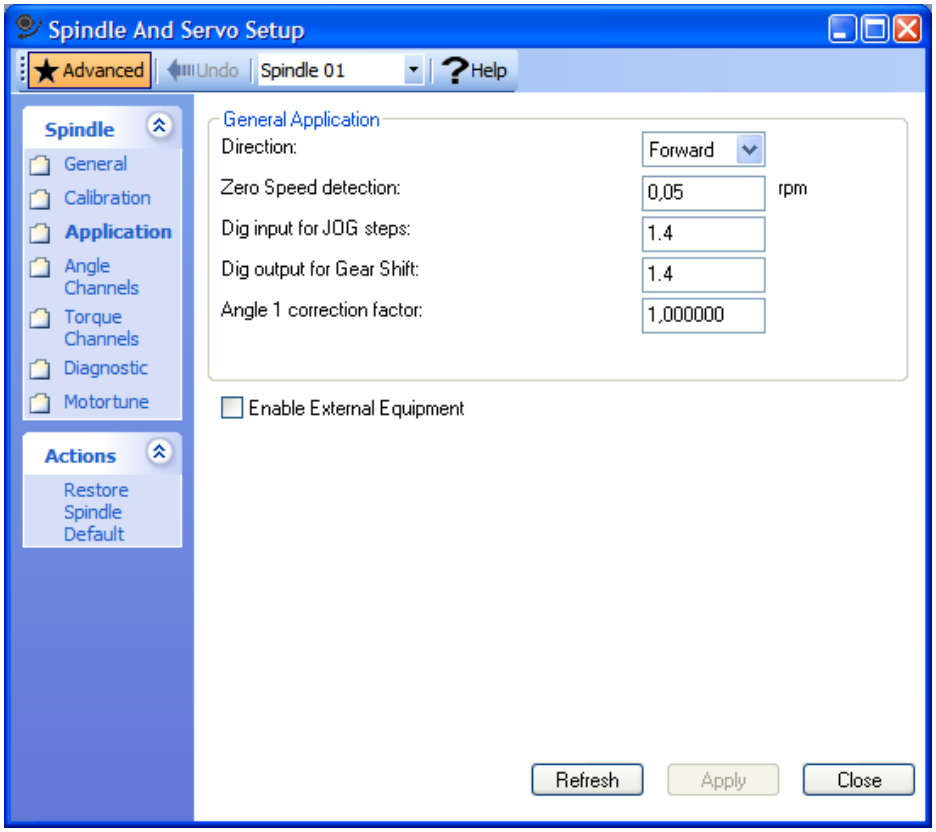
Note that if you change the Gain it is recommended to recalibrate the scale factor.

For the respective angle channel you may use the **Wind Up** to compensate for the torsion in the gears and the shaft caused by the applied torque. They are expressed in the unit [degrees at maximum spindle torque]. Please note that it is the torque transducer used for control that is used as input for the compensation. See chapter Wind Up Coefficient in Calibration procedures in PowerMACSfor how the Wind Up value can be calculated.

The **T/C Correction Factor** is used to calibrate the current measured as torque values.

4.6.3 Application page

The Application page contains application specific parameters, they are stored only in the setup and not on the spindle.



Use the **Direction** field to select the basic direction for the spindle, normally **Forward** (clockwise). If the spindle should use the inverse direction select **Backward** (counter clockwise). You can also control the direction from the station PLC by selecting one of the items **PLC P1** to **PLC P5** which corresponds to the PLCs Station variables output GENOUT_1 to GENOUT_5 respectively. If the selected PLC output is set to 1 then the spindle will run forward (clockwise), otherwise backwards (counter clockwise).

Zero Speed specifies how fast the spindle can rotate, in revolutions per minute, and still be considered having zero speed.

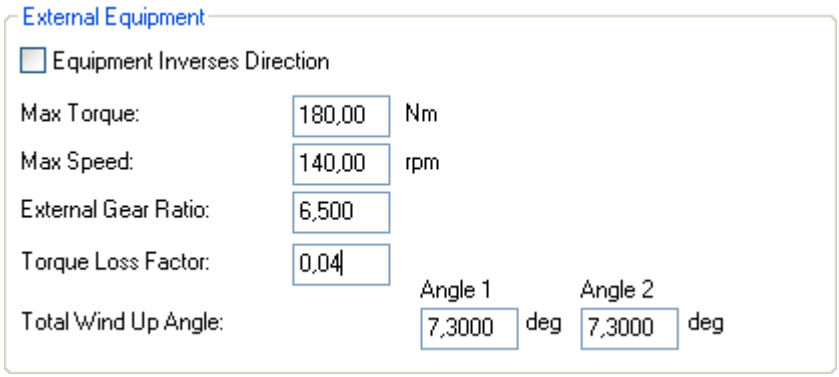
If you run a JOG - Run until digital input goes high / low step, the spindle runs until a specified input goes high or low. Specify in **Dig input for JOG steps** which digital input to supervise. Enter as **<TC no.>.<channel>,** e.g. 1.4. Leave blank if you do not have any input for this. See also the description of I/O Device in chapter: Peripheral Devices.

Specify in **Dig output for Gear Shift** the digital output that is connected to the spindles gearshift unit. This can be controlled using the GS - Run Gear Shift step. Enter as **<TC no.>.<channel>**, e.g. 1.4. Leave blank if you do not have any output for this.

The **Angle correction factor** is used to calibrate the angle sensors, this is normally not necessary and the value should be left as 1.0).

4.6.3.1 Application External Equipment

The **Enable External Equipment** checkbox should be checked when an external gearbox or similar equipment is connected to the spindle. The following frame appears when external equipment is enabled:



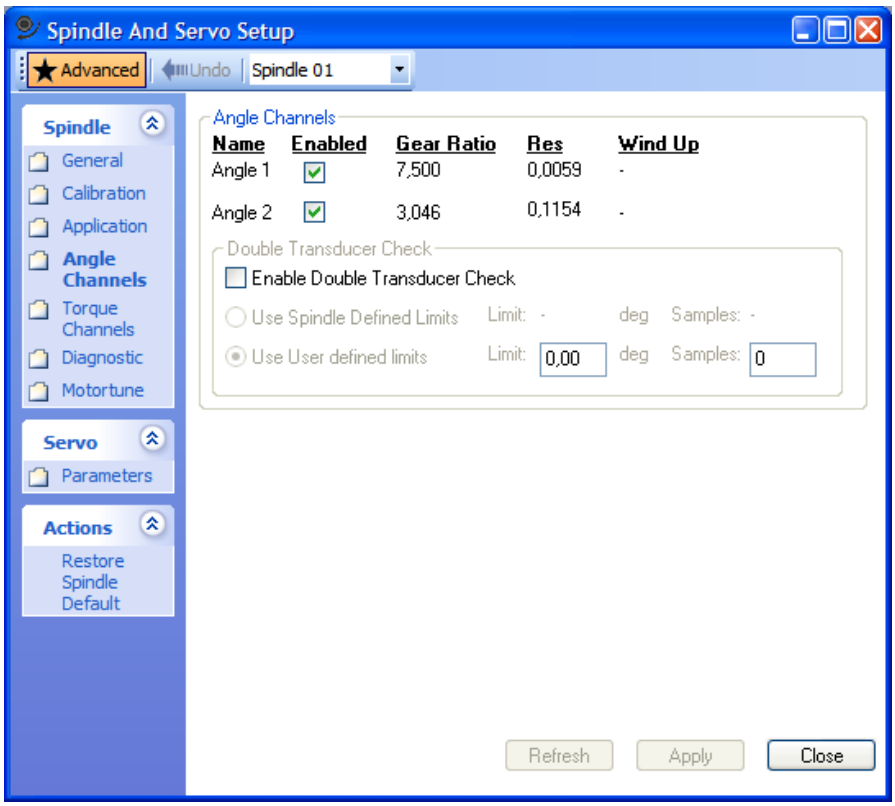
Use the checkbox **Equipment Inverses Direction** if the connected equipment inverses the spindle direction. A new **Max Torque, Max Speed** must be entered together with the **External Gear Ratio.**

The **Torque Loss Factor** is used to compensate from the torque measurement loss induced by the external equipment.

It is also recommended to supply a new *total* **Wind Up Angle** that is valid for the **complete construction** (spindle inclusive the external equipment). For instructions how to calculate this value see chapter Wind Up Coefficient in Calibration procedures in PowerMACS.

4.6.4 Angle Channels page

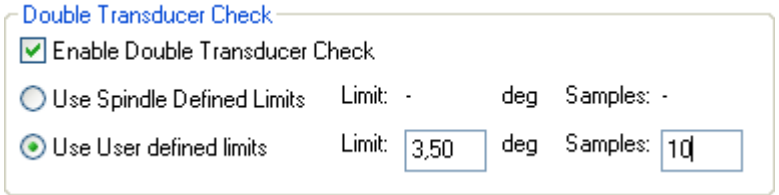
The **Angle Channels** page contain information about the equipped angle channels. It is also possible to configure the double transducer check here.



This page shows basic properties for each equipped angle channel, it is possible to **Disable** the secondary Angle channel (if equipped).

4.6.4.1 Angle Double Transducer Check

It is possible to configure a default double transducer check to run during cycles if two angle channels are equipped on the spindle.



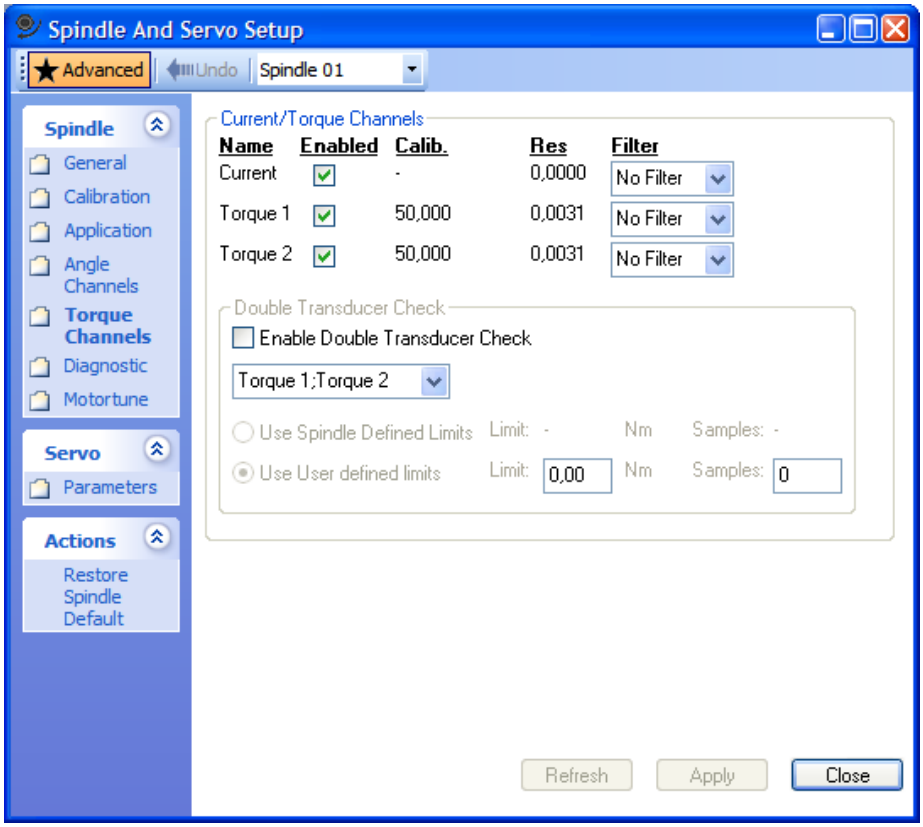
It is possible to select if the limits for the test should be read from the spindle (default provided) or if you want to specify your own limits. The Spindle defined limits are only displayed when ToolsTalk PowerMACS is connected to a system.

It is possible to override the spindle configured transducer check for specific steps by using the Angle Difference restriction.

Note The angle diff check is not executed while the spindle functional test is performed in the D - Diagnostic Step.

4.6.5 Torque Channels page

The **Torque Channels** page contain information about the equipped torque channels. It is also possible to configure the double transducer check here. This page also show information about the current channel.

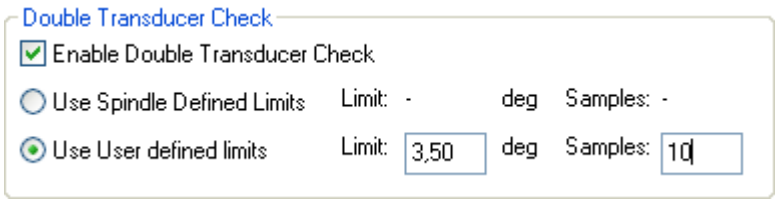


This page shows basic properties for each equipped torque channel (including the Current channel), it is possible to disable any channel here by unchecking the **Enabled** checkbox.

Filter, Hz is the cut-off frequency of the low pass filter used to filter the torque and current inputs. Normally this should be set to **No Filter** but can be specified if a certain application requires it.

4.6.5.1 Torque Double Transducer Check

It is possible to configure a default double transducer check to run during cycles.



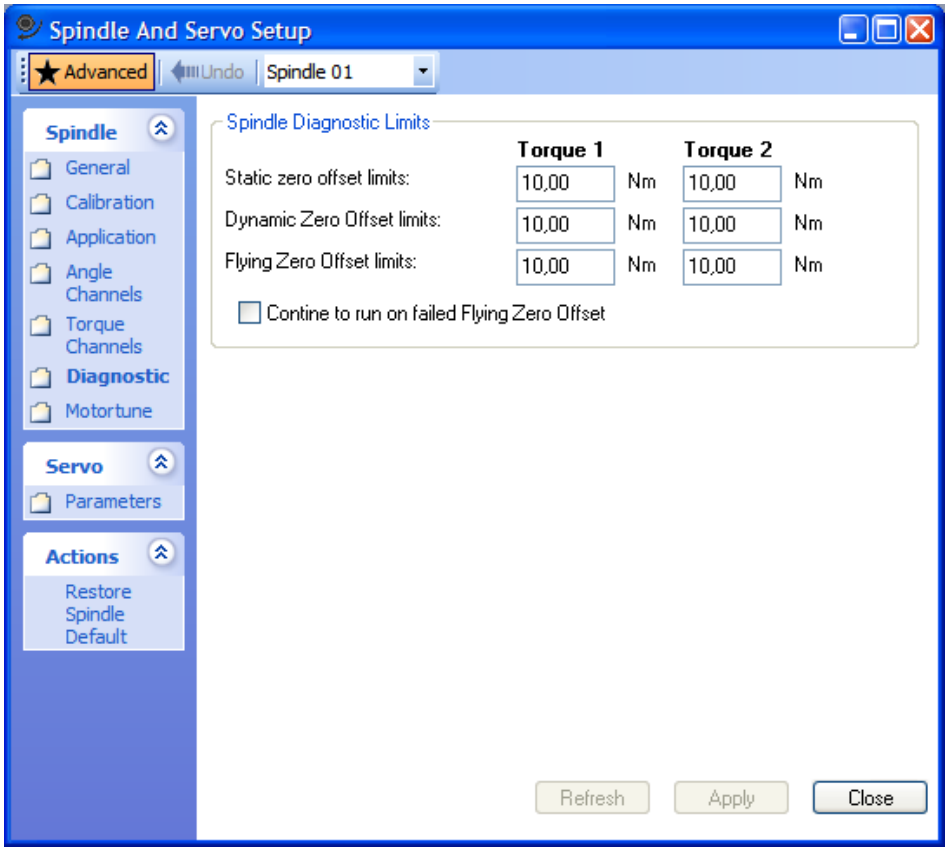
It is possible to select if the limits for the test should be read from the spindle (default provided) or if you want to specify your own limits. The Spindle defined limits are only displayed when ToolsTalk PowerMACS is connected to a system.

It is possible to override the spindle configured transducer check for specific steps by using the Torque Difference restriction.

Note! The torque diff check is not executed while the spindle functional test is performed or zero offset is being measured. Zero offset is normally measured during the execution of a D - Diagnostic Step but can also be done during other steps if Flying zero offset is used (see chapter: Diagnostic).

4.6.6 Diagnostic page

The parameters located on the Diagnostic page specifies the limits of the zero offset tests in the D - Diagnostic Step.



Only the equipped channels are visible. It is also possible to specify if a cycle should continue to run after a failed flying zero offset test.

4.6.6.1 Static Zero Offset

To a large extent you can eliminate torque-measuring errors by performing zero offset check and compensation. When such an operation is done is controlled by the execution of a D - Diagnostic Step.

Static zero offset is measured without turning the spindle. The static zero offset is the average torque measured during **Measuring time**. The absolute value of the static zero offset must be below **Maximum Static Zero Offset**. If not, step status is set to **Fatal** and the error flag STZODIAG is set.

4.6.6.2 Dynamic Zero Offset and Angle Count

Dynamic zero offset is measured as the average torque while the spindle first runs the specified **Target angle** degrees in forward direction and then in the reverse direction.

If an **Angle count test** is specified the motor will run exactly as when performing a Dynamic zero offset test. The angle is measured separately in the forward and the backward direction. Both measured angles must be greater than the specified **Target angle – Angle limit** and less then the specified **Target angle + Angle limit**. If not then step status is set to **Fatal** and the error flag ANGDIAG is set. If Dynamic zero offset is selected as well then it is measured at the same time as the Angle count test.

4.6.6.3 Flying Zero Offset

Flying zero offset is measured as the average torque while the spindle runs forward. The measurement is not started immediately when the diagnostic step is started but first after that the **Time from cycle start to measuring** has elapsed.

The average value is calculated during **Measuring time**. The absolute value of the measured offset should be below **Maximum Flying Zero Offset**.

Whether or not the step is stopped when the offset exceeds this value depends on the value of **Continue to run on failed Flying Zero Offset**. If checked then a too high value just generates a warning (FLYZODIAG) and the bolt is not stopped. If not checked, then the occasion is considered to be an error (FLYZODIAG) and step status is set to Fatal.

Since measuring does not necessarily start immediately when the diagnostic step does, both measuring and evaluation may be done while during the execution of another step. If the cycle ends before measuring is started, that is before **Time from cycle start to measuring** has elapsed, then the measuring is cancelled and neither an error nor a warning is generated.

Note! The measured zero offset value is used to compensate all torque readings from that the next step (compared to the step where the last zero offset measuring was taken) is started.

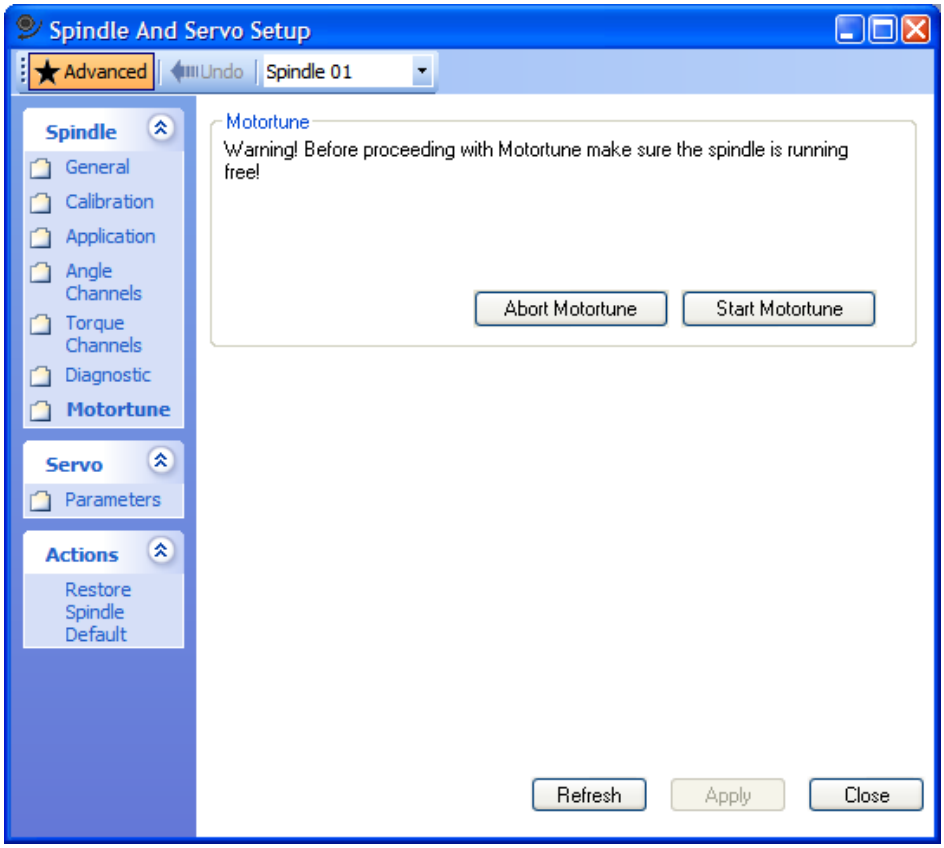
4.6.6.4 Order of execution

The below text describes the sequence of actions that can occur:

```
IF (Static Zero Offset test) THEN
   Measure the Static Zero Offset;
   Save as Zero Offset compensation value if within limits;
ENDIF

IF (Dynamic Zero Offset test OR Angle test) THEN
   First run in the forward direction and then in the backward direction;
   Measure average torque and angle pulses in both directions;

   IF (Dynamic Zero Offset test) THEN
      Check the measured Dynamic Zero Offset;
      Save as Zero Offset compensation value if within limits;
   ENDIF

   IF (Angle test) THEN
      Check that both the measured angles are within specified limits;
   ENDIF
ENDIF

IF (Flying Zero Offset test) THEN
   Run forward a programmable time from the start of the diagnostic step (do not measure);
   Start measuring the torque and run another programmable time;
   Save as Zero Offset compensation value if within limits;
ENDIF
```

4.6.7 Motortune page

If you have a service enabled ToolsTalk PowerMACS it is possible to Motortune a spindle.



Press **Start Motortune** to begin the motortune operation, note that the spindle will run at a very high speed so it is absolutely **critical** it is running free. Motortune status can be observed through events.

4.6.8 Automatic transducer protections

To protect the transducer from overload a number of non-configurable restrictions are continuously executed. The errors produced by these tests are always considered **Fatal**.

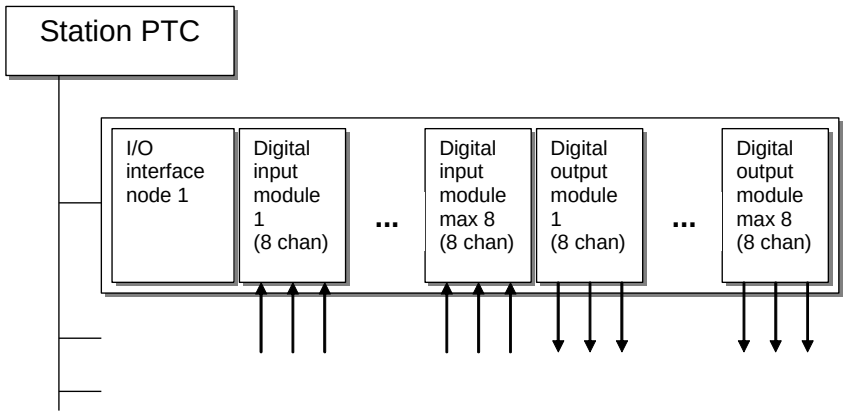
Transducer overload protection (Spindle Protection)

This test supervises that the torque measured on the monitoring channel does not exceed the maximum allowed torque for the spindle. The measured torque must be less than **Max Torque** + 5% and **Max Torque** + 5 Nm otherwise a transducer protection event is generated and error bit TPROT is set in the Bolt level result variables "Errors"/"RM Errors".

4.7 I/O

The I/O form is invoked using menu item **Setup – I/O…**. or by double clicking an I/O device in the System Map.

An I/O device is used to communicate with hardware modules for digital input and output signals. This can be connected to all PTCs that run a station. The PTC communicates with the I/O using a local fieldbus (of DeviceNet type) over its CAN channel.



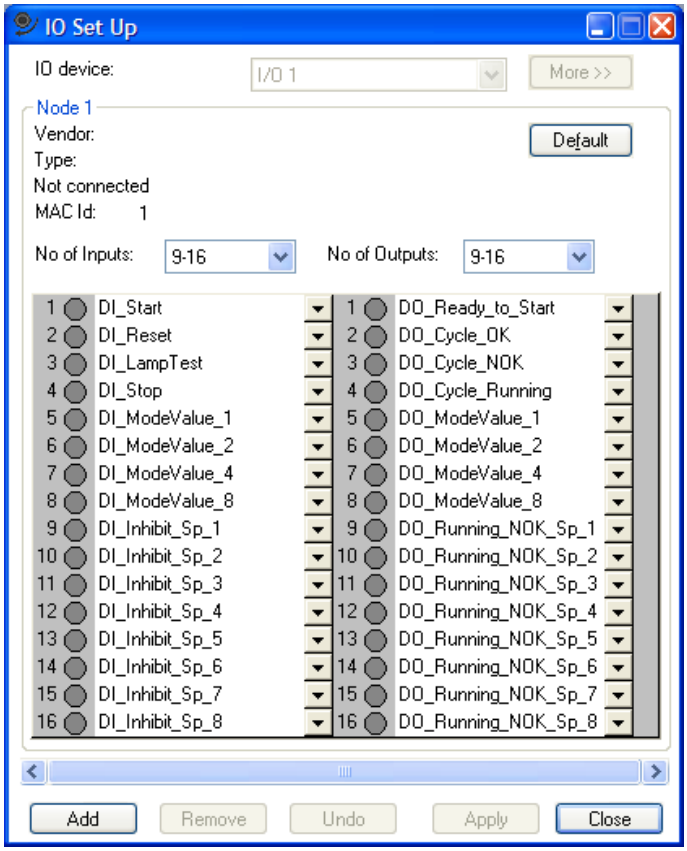
The maximum configuration on the local I/O bus is six nodes. Each node can have a maximum of 64 inputs and 64 outputs. However, the total amount of I/O points must not exceed 192 inputs and 192 outputs per TC.

Currently the following I/O types are supported:

- VIPA DeviceNet slave IM253DN
- Allen-Bradley CompactBlock I/O Modules Series B (1791D)
- Beckhoff BK5210
- Wago DeviceNet I/O modules 750-346

An I/O device can be added to each PTC that has a PLC, i.e. the first PTC in each station.

You can add an I/O device to a station TC using the System Map. If you have used the Set up of a new system it is automatically added.



Press **Add** to add a new node. A new node is added with header but with no inputs or outputs. Press **Remove** to remove the last node. Press **Apply** to make to the change immediately without leaving the form. With the horizontal slider you can select which node you want to present.

Specify for each node how many inputs and outputs you have. Normally nodes are built with I/O blocks, each having eight inputs or outputs. If you have one block, specify 1-8 signals. If you have two blocks specify 9-16 signals, and so on.

Beside every input or output signal there is a combo box. With this you can select to which PLC signal you want to connect the input or output. The I/O device, as well as Stacklight, Indicator Box, and Operator Panel, can map all boolean signals located in the I/O area and in the shared area. Boolean outsignals starting with “SO_” in the fieldbus area can also be mapped by these devices since they belong to the shared signals. See I/O Device for more details on how to set up and program these signals.

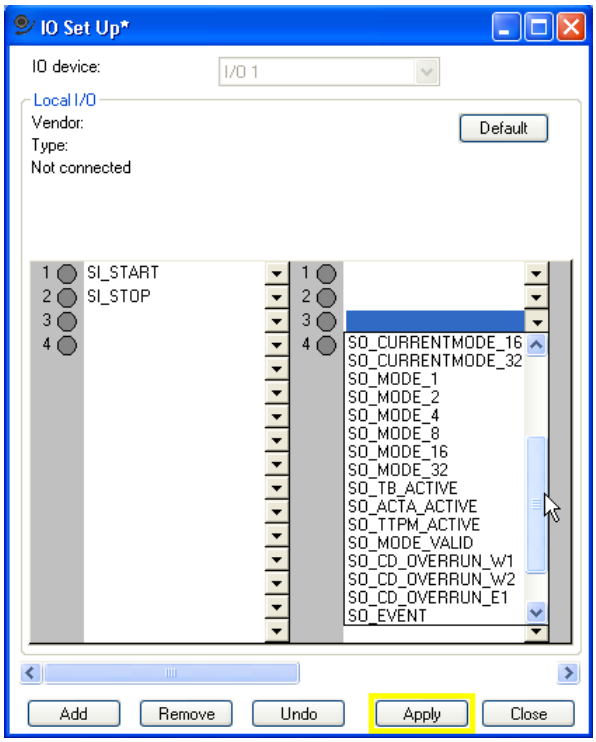
When a node is initially added it is set up with a default configuration.

Node	PLC input signal	PLC output signal
1	Name of signal 600..663, if there is one	Name of signal 800..863, if there is one
2	Name of signal 664-727, if there is one	Name of signal 864-827, if there is one
3	Name of signal 728..791, if there is one	Name of signal 828..891, if there is one
4	No default is set up	No default is set up
5	No default is set up	No default is set up
6	No default is set up	No default is set up

Note: The PLC signals presented in the combo box lists are those that existed when the form was opened. If you open the PLC and change these names at the same time as an IO Setup or IO Map form is open, the change will not be displayed until an IO form is opened the next time. The best way is therefore to first create all names in the PLC, then configure the IO.

4.7.1 Local I/O

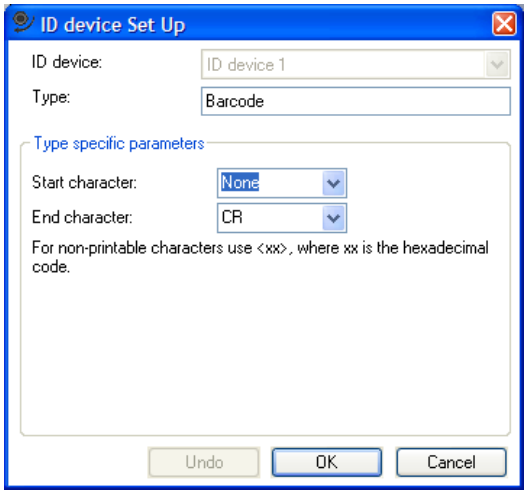
Local I/O is visible as the first node of the I/O device. In order to map the local I/O, add an I/O device to the setup and open the I/O form The Local I/O is mapped by selecting PLC signals in the comboboxes in the form, like for the other I/O nodes.



As default, I/O node 1 is part of the I/O device. For users only needing the Local I/O, pressing the Remove button will remove I/O node 1 from the setup.

4.8 ID device Set Up

The ID device form is invoked using menu item **Setup – ID device…**. or by pressing the **Open...** button displayed in the **Details** window when the selected node in the System Map is an ID device.



Select the ID device to view or configure using the **ID device** combo box.

ID devices are equipment that can read identification information from, and write result data to, the work piece.

If the device is connected to a PowerMACS Station the information read from an ID device may be added to the tightening data, as the station level result variable "Wp ID", or the Multiple identifier result variables "ID Res x". See description of Work piece identifier and Multiple identifiers in chapter: Advanced Station Settings for how to configure this.

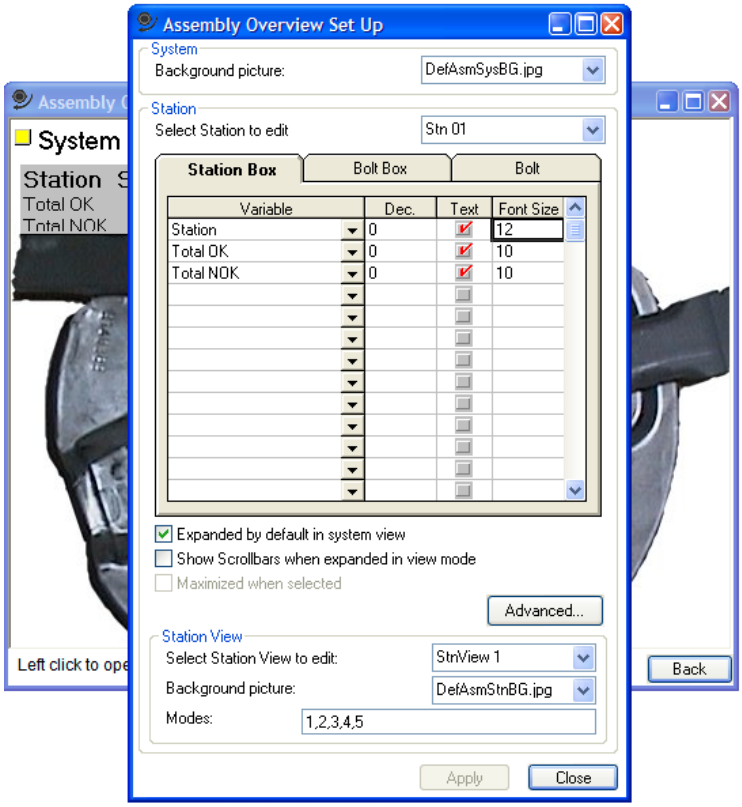
The data read from an ID device is also available to the PowerMACS PLC making it possible to process it as wished.

Note! To create a new ID device or to change the communication settings for a device you must use the System Map.

Depending on the type of the ID device the frame **Type specific parameters** displays different controls. For a detailed description of these see chapter: ID device.

4.9 Assembly Overview Set Up

The **Assembly Overview** displays the current status of the system, its stations and bolts. Which data to display in the different boxes, and their layout, are controlled by the **Assembly Overview Set Up** form. It is invoked using the **Set Up - Assembly Overview…** menu item



When selecting the **Set Up - Assembly Overview…** item two forms are displayed. First the Assembly Overview Set Up form and secondly the Assembly Overview form, now running in Set Up mode.

The first form is used to configure the basic settings for the Assembly Overview, that is which data and pictures to display in the Assembly Overview window.

The second form is used to define the position and size of the different station and bolt boxes.

The Assembly Overview Set Up form

Use the **System Background Picture** combo to select the JPG file to display as System View background.

To include your own JPG files you must copy the files to the directory **Bmp** located in the directory you installed the ToolsTalk PowerMACS application in. All files with extension JPG in this directory are listed in the picture combo boxes when the form is opened the next time.

Note 1: The selected JPG files are not included as a part of the setup. In order to display the same pictures on another computer then the one you created the setup with you must copy the files manually.

Note 2: The JPG files are converted to bitmaps when used by ToolsTalk PowerMACS. The time it takes to load a JPG file, and the amount of memory it occupies, therefore depends heavily on the resolution and color depth of the picture. It is therefore important to keep these as low as possible if using a low end PC.

In the **Select Station to edit** combo box choose the station to configure. Please note that it is possible to have different settings for each station.

For each station you may select which station and bolt level variables to display in the corresponding boxes. For a description of the variables see chapter: Result variables.

Use the combo boxes on the **Station Box** tab to define which result variables to display in the Station box.

The combo boxes on the **Bolt Box** tab controls which of the result variables that are displayed in the Bolt boxes.

Use the combo boxes on the **Bolt** tab to define which result variables to display in the Bolt detailed view.

The variables selected are displayed in the same order as entered in the grid. Use the columns **Font Size**, **Text** (whether to include the prompter text) and **Dec** to control the format of the each variable.

Check **Expanded by default in system view** if the Station box should be expanded, that is show its Bolt boxes on top of the stations background picture (“Station detail view”), when the Assembly Overview form is opened. If not checked, then the Station box is displayed in its collapsed form which means that it only show the variables selected on the Station Box tab, that is no background and no Bolt boxes.

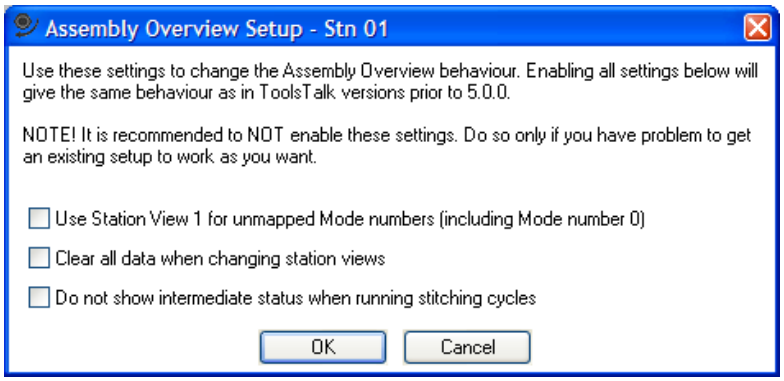
Check **Show scrollbars when expanded in view mode** if the “Station detail view” should display scrollbars at the bottom and to the left of its frame when the Assembly Overview form is displayed in its view mode (that is, when not opened from "Set Up - Assembly Overview"). The scrollbars are always displayed when in set up mode to make it possible to position the picture within the “Station detail view”.

Check **Maximized when selected** if the “Station detail view” should occupy the whole Assembly Overview picture when displayed (that is when the Station Box is clicked on in the System view).

The above listed configurations are common for a given station regardless of which mode the station is running (see The Mode Table form for a description of modes) . However, there are a number of aspects that can be displayed differently depending on the currently selected mode: These are defined as a so called **Station View**. Each station can have up to ten Station Views and for each of them you may specify the following:

- Which **Background picture** to use for the Station detail view.
- The size and position of the Bolt boxes in the Station detail view (this is edited directly in the Assembly Overview form).
- For which **Modes** the Station View should be used. These are entered as a comma separated list of mode numbers.

Click on **Advanced…** to open a new window with advanced settings:



All of these settings are used to change the Assembly Overview behavior in different aspects.

Note! It is **not** recommended to enable these settings. Do so only if you have problems to get an existing set up to work as you want.

Enabling *"Use Station View 1 for unmapped Mode numbers (including Mode number 0)"* will cause Station View 1 to be displayed when a Mode number that does not map to a station view is selected or Mode number is set to zero. The default behaviour is to keep the currently displayed Station View.

Enabling *"Clear all data when changing station views"* will cause all station and bolt boxes to be cleared of its result information and the background color set to grey when the station view is changed. The default behaviour is to keep all result information and background color and use PLC station output variable RESETSTATUS to clear the data.

Enabling *"Do not show internediate status when running stiching cycles"* will cause all bolt boxes to remain yellow until the complete tightening cycle is finished and the cycle data is produced. Default behavior is that the bolt boxes background color is updated with the status of the sub cycle but no result information is shown until the complete tightening cycle is finished.

The Assembly Overview form in Set Up mode

This form is used to define the position and size of the different station and bolt boxes. The layout is modified directly in the displayed picture by using the mouse.

To change the positions of a box move the mouse pointer to the upper left corner of the box. When the mouse pointer changes to a vertical arrow pointing upwards, press the left mouse button and reposition the box.

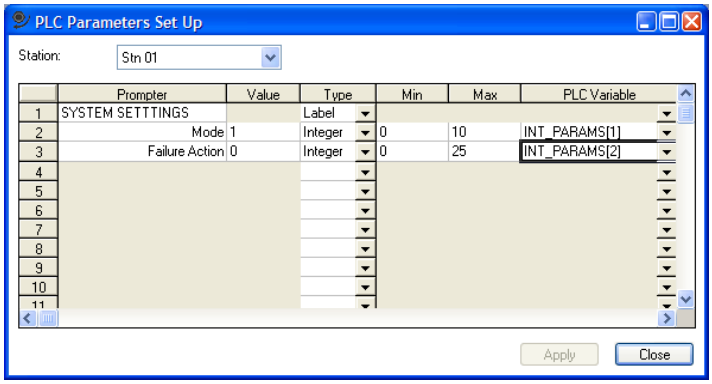
To change the sizes of a box move the mouse pointer to the lower right corner of the box. When the mouse pointer changes to a diagonal arrow, press the left mouse button and resize the box.

All boxes of a given type will have the same size and display the same variables.

When you are satisfied with the settings press the **Apply** button in the Assembly Overview Set Up form to save the settings of the current Station View.

4.10 PLC Parameters Set Up

This form is invoked using the menu item **Set Up - PLC Parameters**.



The PLC Parameters Set Up form is used to define parameters that are available from inside the PowerMACS PLC (see PLC - Overview).

Defining parameters for the PLC here is an alternative to defining them as a part of the PLC application directly (using the Multiprog editor). This is useful if it is likely that the end user will need to adjust them after delivery, or where one wants to make a PLC application that is generic to some extent.

The end user may change the value of the parameters defined here using the PLC Parameters form.

The form contains 190 rows on which either a parameter or label can be defined. Labels are used only to make the layout of the form easier to read, they are not accessible from the PLC.

For each row you must define the following:

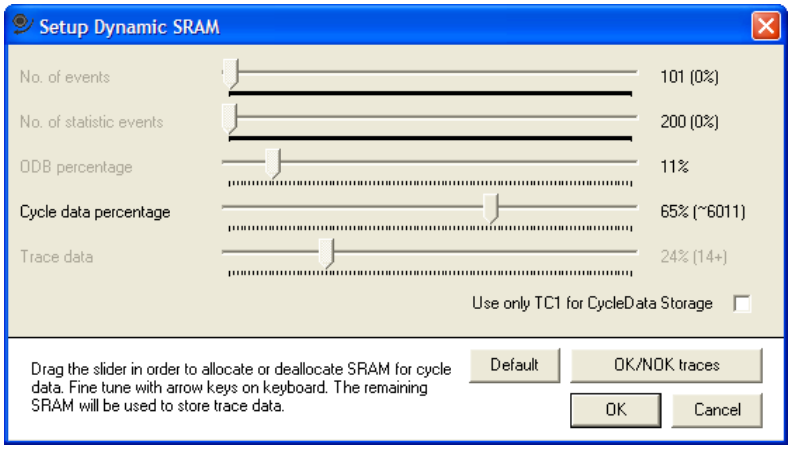
- A **Prompter**: Used to describe the parameter. Max 20 characters long.
- Data **Type** of the PLC variable: One of the following **INT** (16 bit signed integer), **REAL** (32 bit floating point value), **STRING40** (max 40 character long strings), **Label**, or blank.
- **Value**. The default value of the parameters. Not valid if **Type** is blank or **Label**.
- **Min** and **Max**: The min and max value of the parameter. Valid only if **Type** is **INT** or **REAL**.
- **PLC Variable**: Defines to which variable in the PowerMACS PLC this parameter is connected. See also chapter: Station variables).

The maximum numbers of parameters that can be used are:

- 130 of Type **INT**
- 20 of Type **REAL**
- 10 of Type **STRING40**
- 30 Labels.

4.11 SRAM

This form is invoked using the menu item **Set Up - SRAM**.



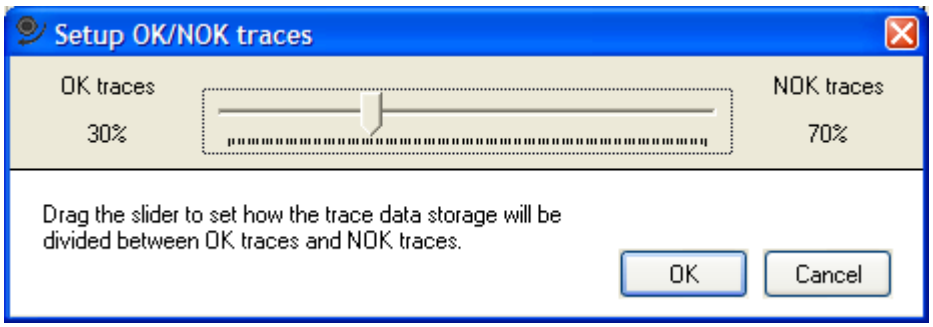
Use this form to set up the configuration of the TC:s SRAM.

Change the size used for cycle data storage by dragging the slider **Cycle data percentage**. The digits shown to the right of each slider is a *guess of the minimum* amount that can be stored given the selected settings.

The checkbox **Use only TC1 for CycleData Storage** is used for backward compatibility, and disables the distributed storage if checked.

Press the **Default** button to reset the settings to their default values.

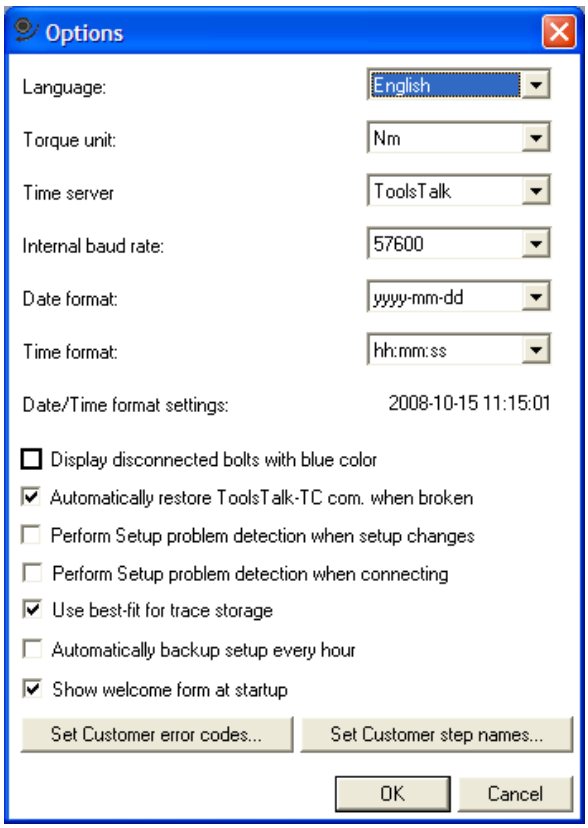
It is also possible to configure the distribution of OK/NOK traces to store. The example below saves 70 % of the memory for NOK traces and 30% for OK traces.



Please see chapter Cycle Data Storage for examples on the number of cycle datas that can be stored on each TC.

4.12 Options

In the Options choice of the **Set Up** menu you can set up your main preferences.



Choose **Language** for presentation on screen. Change will take place when windows are closed and reopened.

Note 1: All texts displayed by ToolsTalk PowerMACS are defined in a so-called Language file. To add support for a new language you just have to copy a file supporting the language in the same directory as your ToolsTalk PowerMACS is installed in. The file should have a name expressing the language it supports and the extension ".LNG", for example "English.LNG".

Use the **Torque unit** combo box to select the unit you want all torque values expressed in. Possible choices are Nm, kNm, FtLbs and InchLbs. This setting can be overridden for a particular device using its reporter.

Use **Time Server** to select from where the clock on the target system will be synchronized. Normally the only option is ToolsTalk PowerMACS but if the system has an Ethernet Protocols device mounted on the System TC then the following alternatives are available as well:

- Open Protocol if the configured Ethernet type is Open Protocol
- FSH if the configured Ethernet type is FSH

Note 3: If you only want to use the Open Protocol device as Time Server you must disable **Collected data** (OK cycles, NOK cycles) in the reporter, otherwise there will be an overflow event as the cycle data queues are never read.

Use **Internal baud rate** to set the communication speed with the **AnyBus CC** module, the following speeds are possible to chose **19200** (default for setups created with versions prior to 7.4.1), **57600** (default speed for setups created from version 7.4.1) and **115200**. The internal baudrate determines the update speed for the connected fieldbus and in most cases the default speeds **19200** or **57600** should be sufficient. Do not chose a higher value for the internal baudrate than necessary since it will affect the overall load of the system, use **115200** only when absolute needed.

Use **Date format** and **Time format** to select the standard for how date/time variables should be formatted when printed. The effect of the selected format is displayed in the Date/Time format settings field.

Check the checkbox **Display disconnected bolts with blue color** if you want the status of all disconnected bolts to be indicated using blue color in the

System Map and Assembly Overview forms. If not checked, bolts that are disconnected OK are colored green while bolts that are disconnected NOK are colored red.

Check the checkbox **Automatically restore ToolsTalk-TC com. when broken** if you want ToolsTalk PowerMACS to automatically try to establish the communication to the System TC whenever it is lost due to transmission errors.

Check the **Perform Setup problem detection when setup changes** checkbox to have the Setup Problem detection tests executed every time the set-up changes. This makes the Setup Problems always displays an up-to-date list.

Check the **Perform Setup problem detection when connecting** checkbox to have the Setup Problems tests executed just before ToolsTalk PowerMACS write the setup to the TC (that is, when the TC is empty or you select to overwrite the existing setup). If the setup contains errors the "Setup Problems" form is shown together with a message suggesting you to correct them before connecting.

Check the **Use best-fit for trace storage** checkbox to make the down sampling of traces try to include peak- and low spot torque/current values at the expense of time resolution. When this option is enabled all peak values and most low spots of torque and current channels are stored. Enabling this option on a 20 second long trace (max) adds up to 90 ms to the cycle time.

Check the **Automaticly backup setup every hour** to have ToolsTalk PowerMACS automaticly store a backup copy of the currently loaded setup every hour.

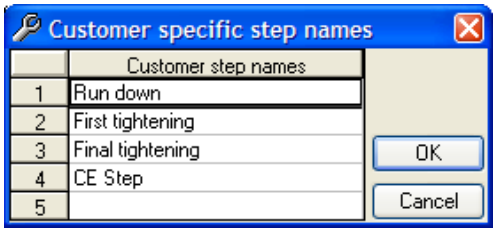
Click on **Set Customer error codes…** to define or edit customer specific error codes, see Set Customer error codes.

Click on **Set Customer step name…** to define or edit customer specific step name, see Set Customer step names.

Note 4: The parameters **Language** and **Automatically restore ToolsTalk-TC com. when broken** are **not** part of the setup. Their values are only stored on the PC from which the parameters are altered. However, since the values are stored in the PC's registry they are remembered between different sessions, also if using different setups. Their respective default values are "English" and "not checked".

4.12.1 Set Customer step names

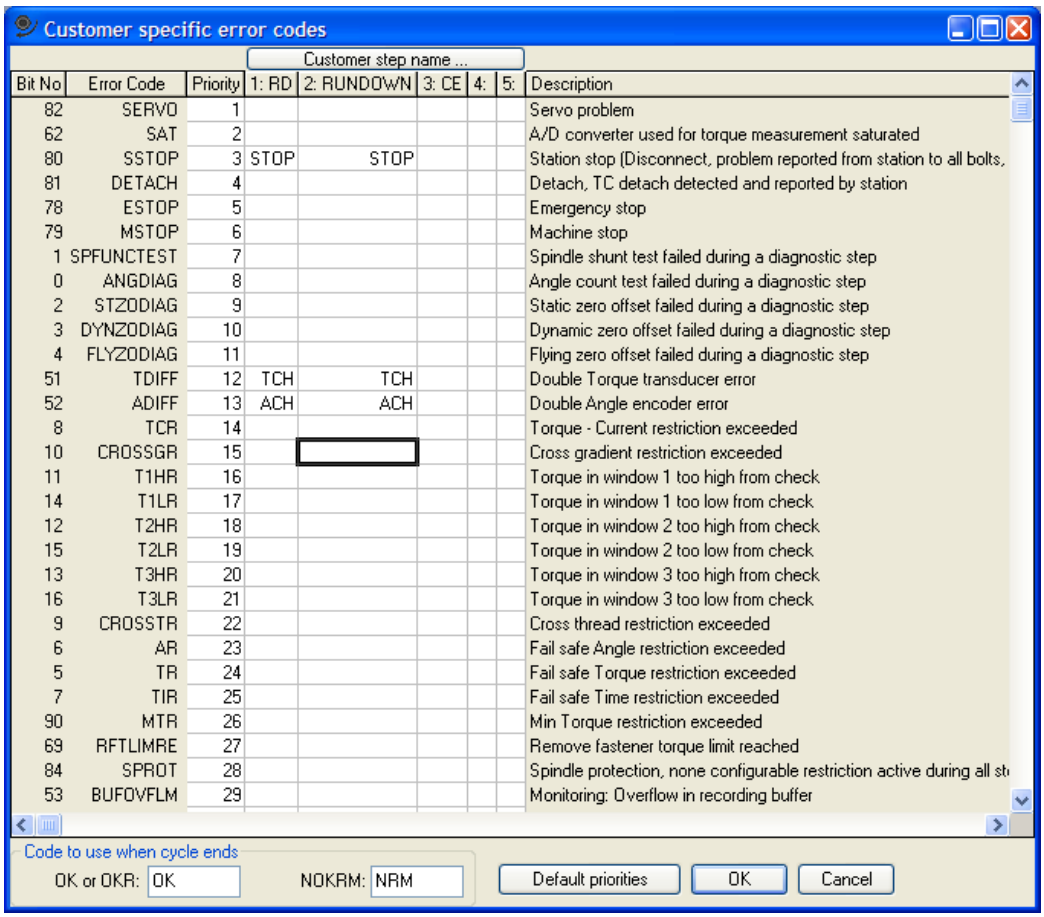
Click on **Set Customer step names…** to display a window where customer specific step names can be defined and edited:



In this window up to five different customer specific step names can be defined. In the tightening program, under Ramps & Other – Other it is possible to select the customer step name to use for that step. The selected customer specific step name is also used to select the customer error code to use for that step, see also Set Customer error codes

4.12.2 Set Customer error codes

Click on **Set Customer error codes…** to display a window where customer specific error codes can be defined and edited:



Bit No	Error Code	Priority	1: RD	2: RUNDOWN	3: CE	4:	5:	Description
82	SERVO	1						Servo problem
62	SAT	2						A/D converter used for torque measurement saturated
80	SSTOP	3	STOP	STOP				Station stop (Disconnect, problem reported from station to all bolts,
81	DETACH	4						Detach, TC detach detected and reported by station
78	ESTOP	5						Emergency stop
79	MSTOP	6						Machine stop
1	SPFUNCTEST	7						Spindle shunt test failed during a diagnostic step
0	ANGDIAG	8						Angle count test failed during a diagnostic step
2	STZODIAG	9						Static zero offset failed during a diagnostic step
3	DYNZODIAG	10						Dynamic zero offset failed during a diagnostic step
4	FLYZODIAG	11						Flying zero offset failed during a diagnostic step
51	TDIFF	12	TCH	TCH				Double Torque transducer error
52	ADIFF	13	ACH	ACH				Double Angle encoder error
8	TCR	14						Torque - Current restriction exceeded
10	CROSSGR	15						Cross gradient restriction exceeded
11	T1HR	16						Torque in window 1 too high from check
14	T1LR	17						Torque in window 1 too low from check
12	T2HR	18						Torque in window 2 too high from check
15	T2LR	19						Torque in window 2 too low from check
13	T3HR	20						Torque in window 3 too high from check
16	T3LR	21						Torque in window 3 too low from check
9	CROSSTR	22						Cross thread restriction exceeded
6	AR	23						Fail safe Angle restriction exceeded
5	TR	24						Fail safe Torque restriction exceeded
7	TIR	25						Fail safe Time restriction exceeded
90	MTR	26						Min Torque restriction exceeded
69	RFTLIMRE	27						Remove fastener torque limit reached
84	SPROT	28						Spindle protection, none configurable restriction active during all st
53	BUFOVFLM	29						Monitoring: Overflow in recording buffer

The customer error code is a four (4) character string that can be included in the bolt level result data. Five different series with customer error codes can be used. The name for each series is specified in the window Set Customer step names.

The series to use is specified in each tightening step on the other tab. It is the series specified in the step pointed to by the result variable Failing Step number that is used for the conversion. Only the errors that occurred during that step is used, not any additional errors that happened during e.g. the termination sequence. It is only the errors that happened during the last execution of the step that is used, each time a step is started all errors for that step is cleared.

Each error code has a unique priority. If several error bits are set it is only the customer error code with highest priority among these that is reported. Even if the customer error code being reported has been left blank it will be reported as it is. The table can be sorted on each column to make it easier to edit.

Customer error codes can also be specified for the two cases when the cycle ends without any error bit set, that is, when the cycle ends with status OK, OKR or NOKRM (see chapter: Result variables - Statuses for a description of statuses). Use **Code to use when cycle ends OK or OKR** and **Code to use when cycle ends NOKRM** to specify these respectively.

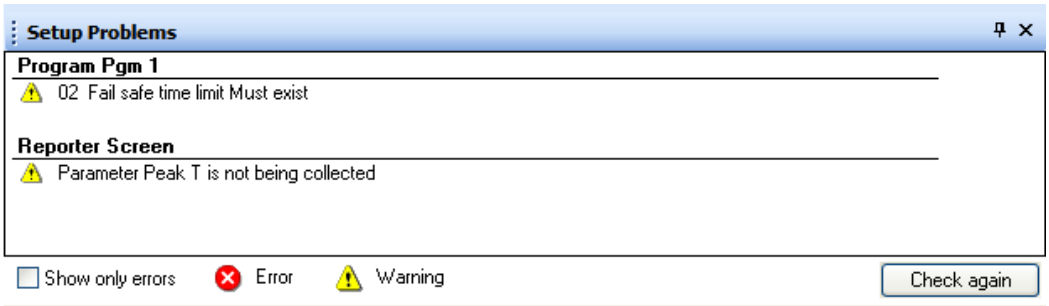
To always be able to determine if the cycle ended OK or not a value must be specified for the OK/OKR case if any error bit has a blank customer specific error code.

Note! **Code to use when cycle ends NOKRM** will be set to the default code used for NOKRM (NRM if the English language file is used) when a setup created with a WinTC version earlier than 5.1.0 is loaded and its **Code to use when cycle ends OK** is empty.

4.13 Setup Problems

The Setup Problems function performs a number of tests on your setup. It will test all combinations of programs against the bolt/spindles that you have setup using The Mode Table form. All target values will be tested with respect to max values defined for the spindles used.

The window is by default docked on the left side of the ToolsTalk PowerMACS main window.



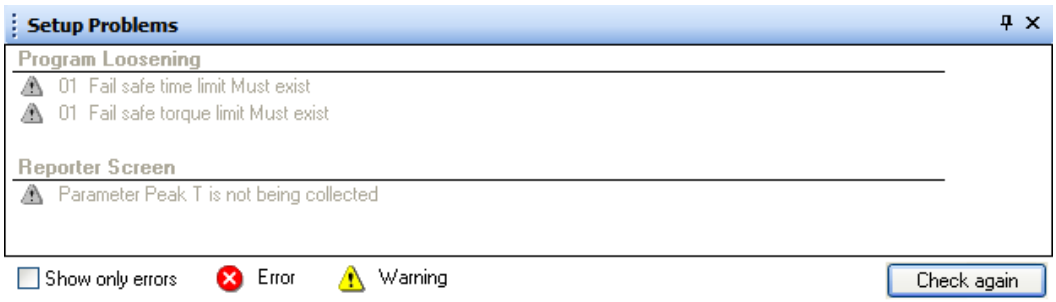
The list contains different categories. For each category a number of messages are displayed. Every message is a classified as either an error or a warning.

If **Show only errors** is checked, only the messages classified as errors are displayed.

Note that the window is displayed even if the test succeeds. If so the text "No errors or warnings found" is displayed.

You can configure the tests to be executed automatically when the setup is modified and/or when connecting. This is controlled from the Set Up Options form.

If the checkbox **Perform Setup problem detection when setup changes** on the Set Up Options form is not checked then setup problems will change appearance whenever the setup is changed to indicate that the last test is no longer valid.



4.14 Table Export and Import

A *setup* is a group of data that completely describes a PowerMACS system, how it looks and how it should work. It can be handled and reused as a complete unit as described in the chapter: “Setups and How to handle them”.

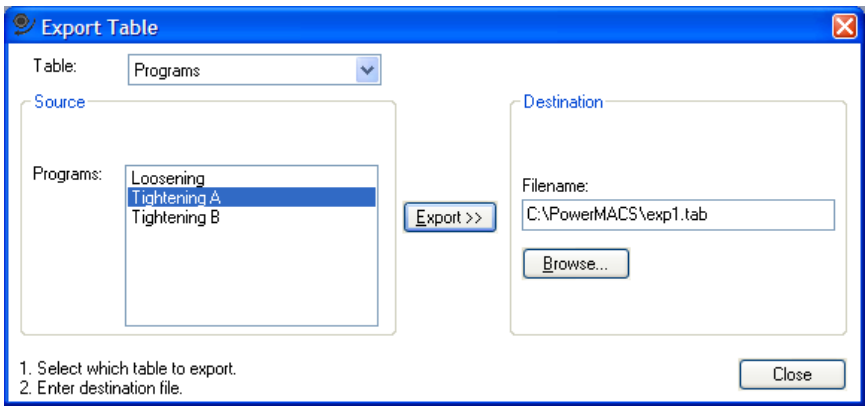
However, it is also possible re-use smaller parts of the setup. These parts are called tables and a table normally represents one particular object in the setup, for example a bolt, a reporter, etc.

A table can be moved from one setup to another by using the Table Export and Import functions by first exporting it from the first setup and then importing it into the second one.

The following tables are available for Export and Import:

- Program
- Sequence
- Station
- Reporter
- Misc (Easy View and SPC configurations)
- Bolt
- Spindle
- Servo
- Device

The **Export Table** form is invoked using the **File - Export** menu item**.**



First select which type of table to export using the **Table** combo.

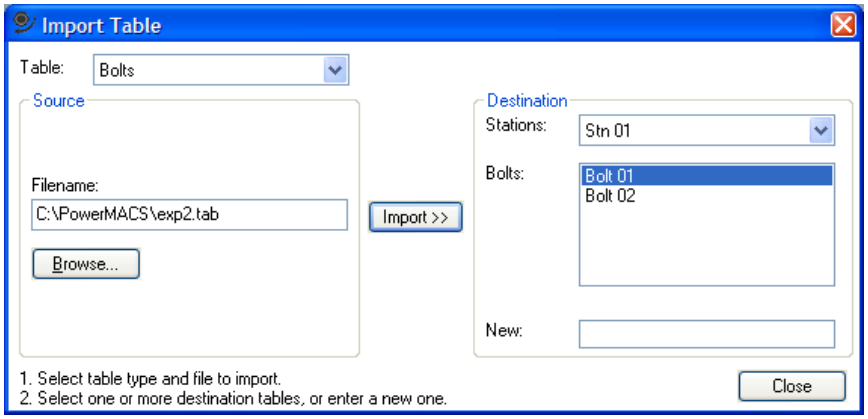
Depending on Table type you specify exactly which instance to export using the displayed controls (**Station**, **Program**, **Spindle** etc.) in the Source frame.

Use the Destination frame to specify to which file the table should be stored.

Finally press the **Export >>** button.

Repeat this for all tables you want to export.

Use the **File - Export** menu item to display the **Import Table** form.



First select the type of table to import using the **Table** combo.

Use the **Source** frame to specify which table file to import.

Specify in the **Destination frame** exactly to which instance you want to import the data. For some table types you may also create a new instance using the imported data. In these cases the **New** field is enabled.

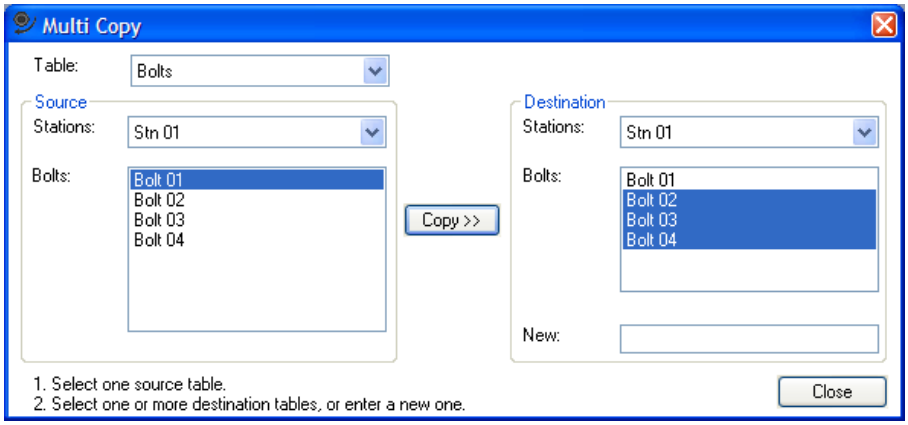
Finally press the **Import >>** button.

Repeat this for all tables you want to import.

4.15 Table Copy

This form is opened using the menu item **Edit - Multi Copy**.

It enables easy copying of a table instance to one or more other instances of the same type.



Use the **Table** combo box to select which type of table to copy. Depending on type of table you can then in the **Source** frame specify which table you want to copy.

Use the fields in the **Destination** frame to specify where to copy the table data. For most table types you may select multiple destinations (if displayed in the list). To do this, hold down the CTRL key and then click on all items you want, or if they are contiguous, mark the first one, press the SHIFT key, and then mark the last one

4.16 Handling of the setup in the target system

The setup consists of all the configuration objects stored in the database along with the PLC program including the PLC source code.

When downloaded to target system this setup data is stored on the System TC. It is from this TC all other TCs in the system access the data they need. Any change made using ToolsTalk PowerMACS while the system is on line directly notifies all dependent nodes about the change. This solution has the advantage of making all TCs but the System TC directly replaceable, without updating them with the latest setup manually.

A backup copy of the setup is always kept on TC 2 in a system in case the System TC needs to be replaced.

4.16.1 Backup of the setup

The purpose of the backup function is to have a fresh copy of the setup available on another TC in the system in case the System TC should need to be replaced. When the System TC is replaced it will then read the backup copy from TC 2 and thereafter start as normally.

The backup function mirrors the setup stored in the System TC to TC 2. The function is automatically enabled in all systems that consist of two or more TCs. Whenever a setup is modified, or completely downloaded to TC1, either from ToolsTalk PowerMACS, or some other remote device (for example the PowerMACS API) the data is automatically copied (mirrored) to TC2.

During a cold start of the System TC checks the status of its own local database and the backup copy stored on TC 2. Depending of their respective status the following cases can arise during system start-up:

Case	Status of setup in the System TC	Status of the backup in TC 2	Action
1	Empty	Empty	Only the System TC is started. A setup must be downloaded using ToolsTalk PowerMACS.
2	Empty	Ok	First the backup setup is copied from TC 2 to the System TC. Then the system starts using the copied setup.
3	Ok	Empty	First the setup is copied from the System TC to TC 2. Then the system starts using setup originally stored on the System TC.
4	OK	OK and identical to the setup in the System TC	The system does a normal start.
5	OK	OK but **not** identical to the setup in the System TC	**The system is not started**. This situation is indicated to the operator in the following two ways: • On the System TC the display will say backup setup differs. • When trying to go connect with ToolsTalk PowerMACS it will refuse to connect and only display the "Resolve Procedure" form describing the situation. See chapter: Replacement of TCs below, for how to resolve the ambiguity.

4.16.2 Replacement of TCs

As described in the table in chapter: Backup of the setup there is only one case where it is not clear to the system which setup to use. That is when both the System TC and TC 2 have valid but different setups stored.

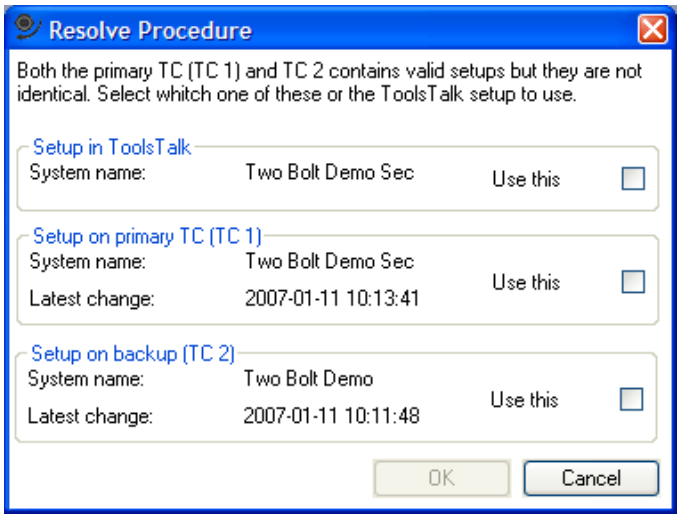
When this situation is detected the system will not start automatically. Instead the System TC will indicate the situation by showing a message that the setups are different on the display.

To avoid this ambiguity you should make sure that the setup stored of the new TC is cleared before it is inserted in the system, especially when replacing the System TC or TC 2.

Clearing the setup stored in a TC is done using the TC display.

It is also possible to resolve the setup ambiguity using ToolsTalk PowerMACS or the TC HMI.

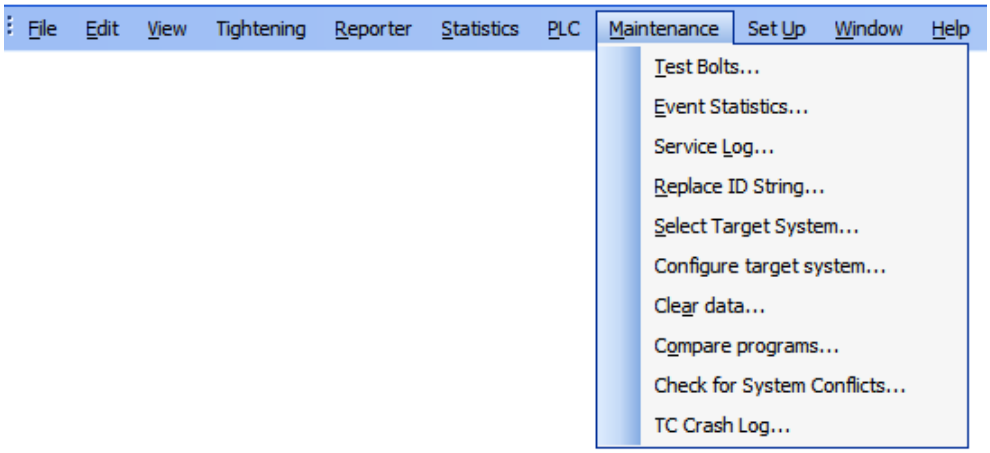
Using ToolsTalk PowerMACS the following form will be displayed when an attempt to connect to the system is made:



The form shows the system name and date and time for the last update of the individual setups and let you choose which one to use. If the setup on ToolsTalk PowerMACS is selected then it will be downloaded to the System TC and the backup will be updated to correspond to it.

4.17 Maintenance

The **Maintenance** menu contains functions for maintenance, like checking the status of the system:



Test Bolts… opens a form from where you can run test passes for one or more bolts of a station. This is an alternative to the PowerMACS PLC when it comes to start tightening cycles. It is also useful for calibration of the torque, current, and angle measuring, see **Calibration**.

Event Statistics… gives you an overview of the most frequent alarms.

Service Log… opens a logbook. Use it to log all your service actions.

Replace ID String… displays a form from which you manually can enter ID strings. Useful if your ID device does not work or just for testing.

Use **Select Target System…** to choose which PowerMACS system to connect to.

Configure Target System… opens a form from where you can configure the TCs of your PowerMACS system. This involves function for upgrading the System Software, changing IP-address settings, etc.

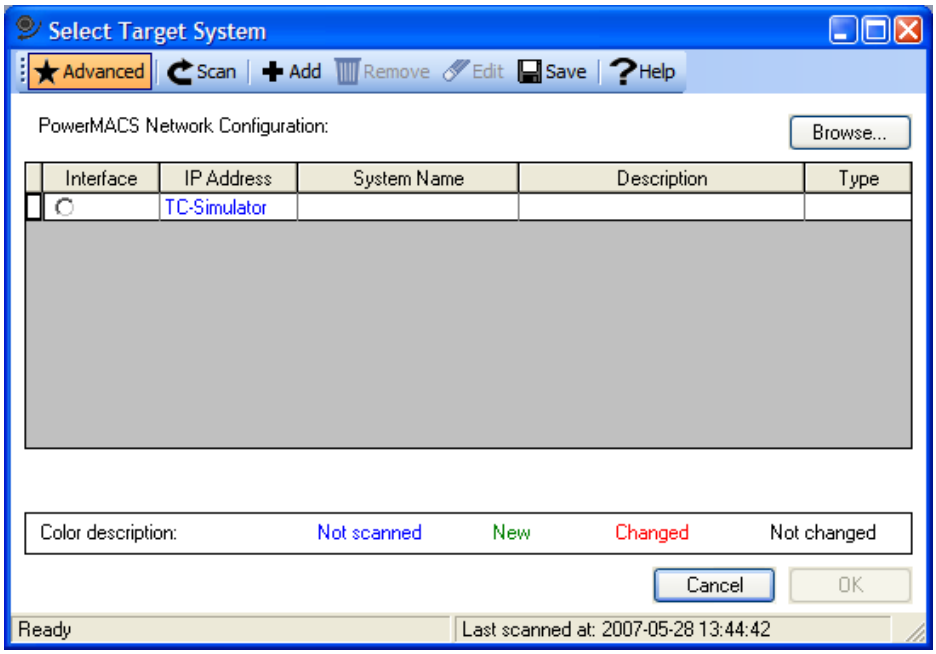
Clear data… is used for clearing data stored in the non volatile RAM of the TCs.

Chose **Check for System Conflicts...** to get an overview of TC set up errors, like mixed versions of TC system software, missing TCs, etc.

TC Crash Log... opens a form that can be used to retrieve debug information from one or more TCs that crashed due to internal, unrecoverable, software errors.

4.17.1 Select Target System

Selecting the **Maintenance - Select Target System** menu item or **Select Target System...** from the **Connect** toolbar drop down menu opens this form that is used to select which target system ToolsTalk PowerMACS should connect to when going online.



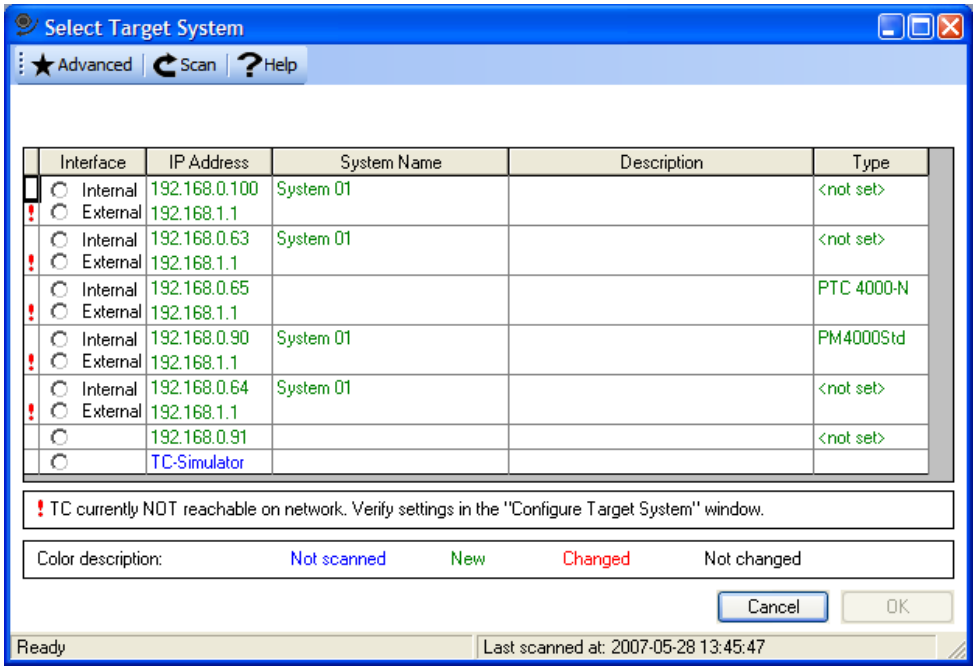
Click **Browse...** to select the PowerMACS Network Configuration file that best describes the network to connect to.

A PowerMACS Network Configuration file describes the expected layout of a PowerMACS Network in the sense that it lists a number of PowerMACS system nodes (System TCs), that for some reason are logically grouped together. Each PowerMACS system in the network is identified with the IP-address of its System TC and free textual description. The main purpose of this file is to document a configuration where several PowerMACS systems are used, for example a complete assembly line.

The selected file will be opened automatically the next time you open this form, even between ToolsTalk PowerMACS sessions. If no file has been selected, a default configuration is displayed.

How to create a Network Configuration file is described later in this chapter.

The network can also be scanned in order to automatically find all PowerMACS systems physically connected to it. Click the **Scan** button to bring up a list of all System TCs on the network.



All System TCs found during scan are compared to the contents of the currently selected Network Configuration file. Depending on the differences the rows in the list is marked with a color.

- Not scanned (Blue) – The TC in the list was not found on the net. When a network configuration file is opened, all rows are marked blue.
- New (Green) – A TC that was not in the list was found on the net.
- Changed (Red) – The attributes of the found TC are different compared to the previously known. Note that the **Description** field is not compared since it is not present in the TC it self.
- Not changed (Black) – The attributes of the found TC is the same as the previously known.

Select the PowerMACS system to connect to by clicking one of the radio buttons in the **Select** column. ToolsTalk PowerMACS does only communicate with the System TC (first TC in a system) so therefore only one selection need to be made.

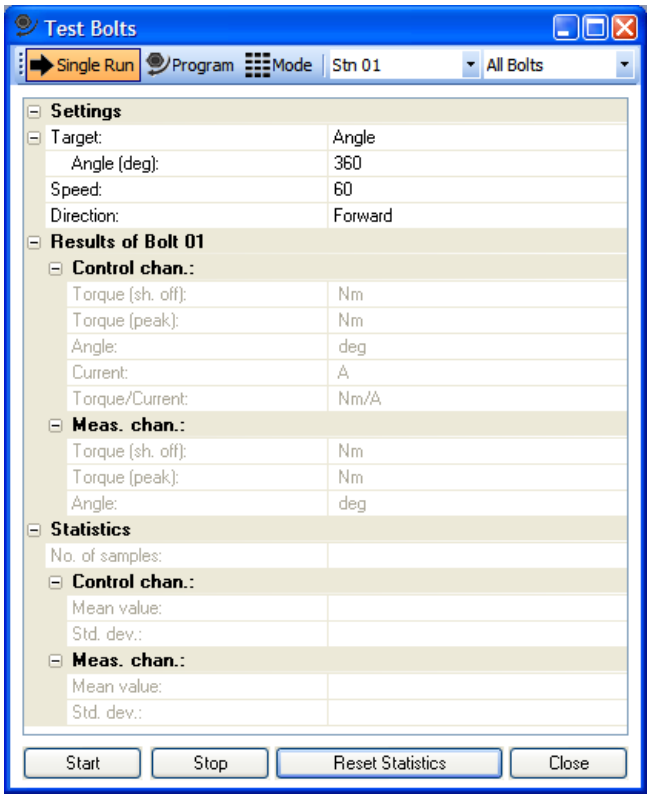
Choose **TC-Simulator** if you do not have a real target system to connect to but still want to demonstrate the on-line functions of ToolsTalk PowerMACS.

TCs that are not possible to connect to due to network settings are indicated with a red exclamation mark. The type of problem is not displayed in detail in this window. It is however possible to select such a TC in this list but before ToolsTalk PowerMACS can connect to the system; the problem must be corrected. Correcting such a problem can be done by using the “Configure Target Systems” window.

To create, or edit, a PowerMACS Network Configuration file use the **Add...**, **Edit...** and **Remove** buttons to insert, change and remove entries in the list. When done editing, save the current configuration to file using the **Save** button. Since the file name is used to identify the network you should give the file a descriptive name. Editing the file can be done without being connected to the network.

4.17.2 Test Bolts

The Test Bolt function gives you an alternative way to the execution of a tightening cycle, running all or only one selected bolt for a station.



First chose which station to test using the **Station** combo box in the toolbar.

Depending on the status of the selected station the Test Bolt form may be disabled. The form is disabled for this reason if:

- The hardware Emergency input is inactive for the TC that executes the station (PLC station input EMERGSTOPIN True).
- The selected stations PLC output DISABLE_TESTBOLT is set True. See also the description of the Station variables in the PLC section.

In either case an explanation of why the Test Bolt form is disabled is displayed at the top of the form.

With the **Bolt** combo box you may select a single bolt or **All Bolts** within the station. Note that if **All Bolts** is chosen and some bolts use the same spindle only the first occurring bolt for that spindle will be run and an error message will be generated.

Should the Test Bolt form be closed, or ToolsTalk PowerMACS be disconnected from the TCs, while a cycle is executing then the cycle is terminated using Machine stop.

Note! When running a station from the Test Bolt form the PowerMACS PLC outputs MODE and BOLTCONTROL are overridden. This means that PLC inputs that depend on these, for example BOLTSTATE, may have values that do not correspond to the values of MODE and/or BOLTCONTROL. To detect these situations in the PLC use the input CURRMODE and TESTBOLT_ACTIVE. The first one always reflects the mode currently known by the station and the second one is set True whenever the Test Bolt form is opened against the station in questions. Also, when the Test Bolt form is closed, or you switch to another station, the station is updated with the value of the PLC outputs MODE and BOLTCONTROL.

You can run the test in **Single Run** mode, in **Program** mode or in **Mode** table mode. You select which by activating the corresponding button.

Single Run mode

In **Single Run** the bolts are run once to a specified **Target** defined either by **Torque**, **Angle** or **Current**. Use the controls in the **Run conditions** frame to define the direction and speed to use when executing the step.

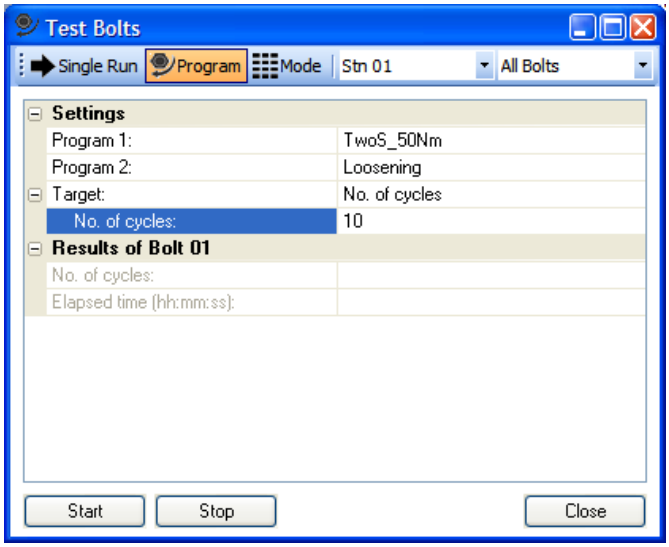
Speed can also be specified as target. If so, then the bolt(s) will run continuously with speed and direction as set by the Run conditions.

When target is reached, or once a second if running speed, a number of measured values are displayed in the **Result** frame. Except for the peak torque value all values are sampled at the switched off occasion.

Note! See chapter: Calibration for how to use Test Bolt to calibrate your system.

Program mode

In Program mode you run the bolts by using one of the existing tightening programs.



Here you can specify how many cycles you want to run, or for how long. If you want to run more than one cycle you will probably need to specify a program to use for loosening (**Program 2**). How to create a new program is described in chapter: Create a new Tightening Program.

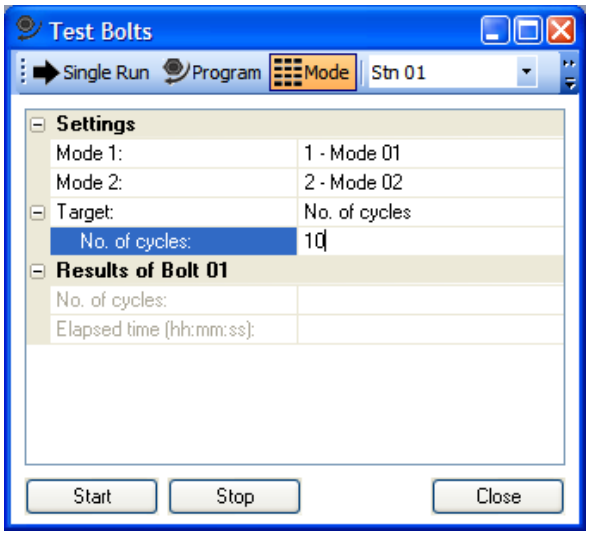
Press the **Start** button to start executing the cycle, Program 1 will be used first and then if **No. of cycles** is > 1 it will continue to alternate cycles between **Program 1** and **Program 2**. Press **Stop** to stop any on-going execution.

Running a program produces Cycle Data and Statistics just as if the bolts where executed by the PowerMACS PLC. This means that data is reported over the normal channels, SPC is calculated, and so on. However, none of the variables that depend on outputs from the PowerMACS PLC are updated. This includes the following variables:

Station level variables	Bolt level variables
Wp ID	Total
Mode	Total OK
Mode No	Total NOK
Free Str	Total Type
Free Str 2	Total Type OK
Free Str 3	Total Type NOK
Free No 1	
Free No 2	
Total	
Total OK	
Total NOK	

Mode Table mode

In Mode Table mode you run the bolts by ordering the station to run a particular mode.



This means that which bolts that will run and which programs they will execute is decided by the current Mode Table (see The Mode Table form).

Use the **Mode 1** and **Mode 2** combo boxes to select which Modes to run. If specifying more than once cycle the station will alter between **Mode 1** and **Mode 2**, starting with the first. Please note that the two combo boxes are filled with the modes that are relevant with respect to the currently selected Bolt. If **All** is selected they will include all modes, without **Program from PLC**, of the station. If a specific bolt is selected they will only include the modes for which the bolt has a program.

Note! It is not possible to use Test Bolts to run modes that use dynamic program selection from PLC. This means that for the selected mode no program may be **Program from PLC** for any bolt in the mode table. Program from PLC requires BOLTCONTROL from PLC, which is overridden by Test Bolts. Only modes without **Program from PLC** are selectable in Test Bolt.

All other functions are the same as for the Program mode.

4.17.3 Calibration procedures in PowerMACS

4.17.3.1 General

It is strongly recommended to perform a calibration of the spindles after the system has been installed at the production site. It is also recommended to make a new calibration periodically (for example one time a year, every 6 months etc.) and this is normally specified in the customer quality control system.

There are at least four alternative ways to perform a calibration:

1. Run on the actual joint.
2. Using a test joint where the test joint should have similar characteristics as the actual one.
3. Using a dummy joint that always is pre-tightened.
4. The manual method.

Method 1: is recommended and is normally giving the most accurate results. One of the other methods can be used, as it is not always possible to use method 1 due to mechanical or similar reasons.

Method 3: The joint is not a real screw joint. It could be a mechanical part with a welded bolt head or similar.

Method 4.1: Program a torque x Nm and the speed 0 (zero) rpm.

- Apply a torque wrench to the inline transducer on the spindle square drive.
- Start the test.
- Turn the torque wrench until the spindle shuts off (spindle torque is released).
- Register the torque value.

Method 4.2: The same as for method 4.1 except that you don't use an inline transducer but only a torque wrench. **This method is NOT acceptable.**

The easiest way to perform a calibration is to use the Test Bolts function. To be able to run spindles from the "Test Bolts" the PLC output "DISABLE_TESTBOLT" must be set False. (In a system running in production this output is normally True for safety reasons).

When running a spindle from the "Test Bolts" menu most PowerMACS PLC functions are not active.

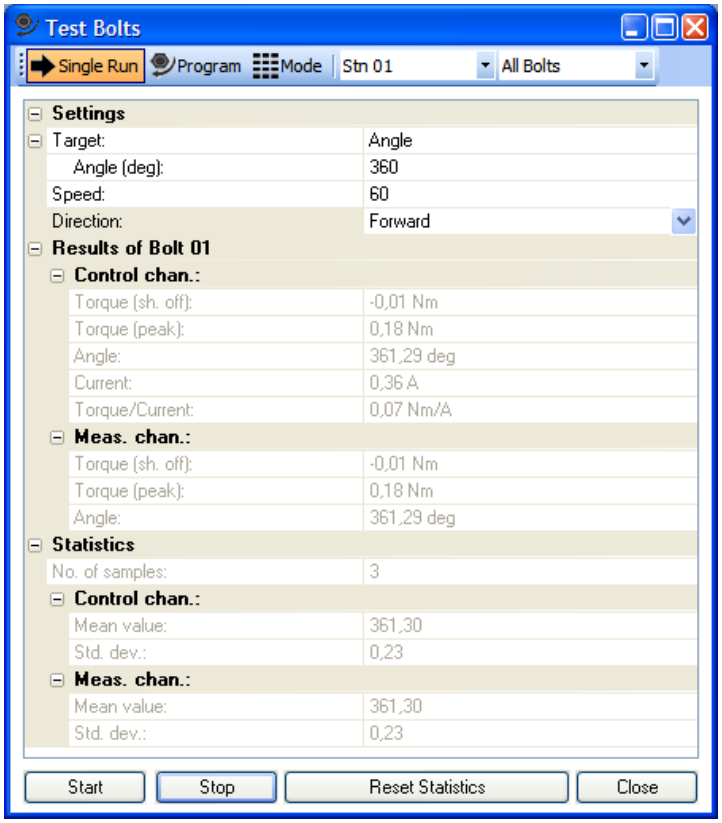
There are three alternative ways to run a spindle from the Test Bolts menu:

- Single run
- Program
- Mode

These are described in the following chapters.

Note! You should only run one bolt (spindle) at a time when calibrating.

Single Run



When you use this function, you will automatically get some statistics directly on the screen. Do the following:

- Select a station.
- Select a spindle.
- Select a speed. The same speed as you have in the final tightening step in the normally used program is a good choice.
- Select the parameter you want to calibrate (Torque/Angle/Current) and a target value.
- Reset the statistics.
- Click on the Start button.

If you use the Stop function, the spindle will stop and a Machine Stop will be generated in the Event Log.

Note! When you use the “single run” function, there is no “zero-offset check and compensation” made. This must be done manually by first running a tightening set that has a D - Diagnostic Step as a first step. The “zero-offset” measured by it will automatically be stored and used until you either run a new Diagnostic step, make a change in the Spindle Set Up form, or power off/on the TC. Use the run “Program” alternative to do this.

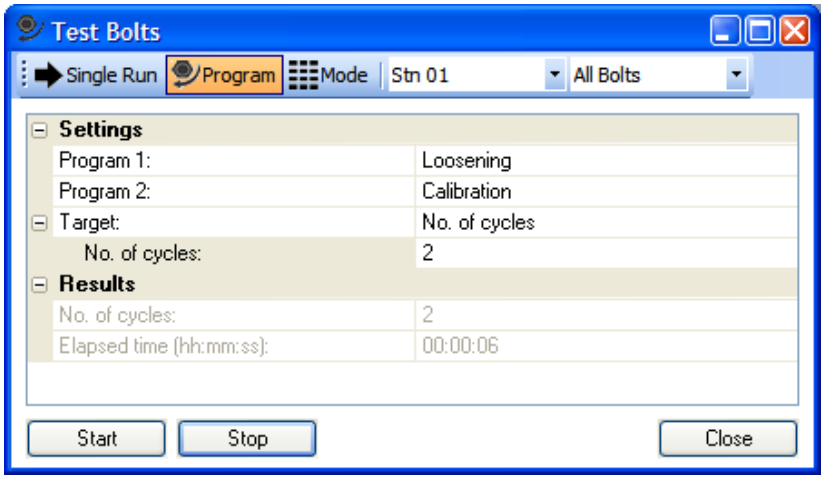
Program

You can either use an existing program or you can create a special tightening program called “Calibration”. The tightening program shall have the following structure:

- Step 1: “Diagnostic” step with spindle functional test enabled and **Spindle Only** selected as zero offset compensation.
- Step 2: Tightening step, e.g. torque xx Nm; speed yy rpm
- Step 3: CE

It is practical to also create a loosening program to be used to loosen the tightened joint.

In the picture below is it defined one tightening and one loosening program.



With number of cycles set to the value 2, a loosening will first be done followed by a tightening.

Mode

Using the Mode option is similar to the Program one. The difference is that in the Mode option it is possible to run different tightening programs for different spindles. As we normally recommend that you shall only run one spindle at a time when you make a calibration is the mode alternative probably not a real option.

4.17.3.2 Torque

All Atlas Copco torque transducers are factory calibrated to a sensitivity within +/-0.3% of the nominal torque. Due to component tolerances in the TC electronics a fine tune of the torque sensor is recommended. Exchange of a torque transducer to an identical unit does not require a new calibration, but for quality control reasons it is anyway normally done.

An inline slip-ring torque transducer and a peak hold monitoring amplifier shall be used for the torque measurement, e.g. Atlas Copco IRTT transducer with ACTA.

Single torque transducer

- If you are using the “Single run” option, run a program with a “Diagnostic” step with spindle functional test enabled to get a new zero offset reading and compensation from the spindle.
- Make at least 10 tightenings to the desired final torque and record the results.
- Calculate the mean torque (X_{mean}) and the difference (X_{diff}) from the set torque value (X_{set}).

  $X_{mean} = \Sigma X_n / n$ where “n” is the number of tightenings
  $X_{diff} = X_{mean} - X_{set}$

- Adjust the spindle parameter **Scale factor** for channel Torque 1 (T_s) (see Spindle Set Up, Calibration) with the calculated value. Increase the **Scale factor** in order to decrease the true torque and vice verse.

  $T_{s\text{-}new} = T_{s\text{-}old} \times (1 + X_{diff} / T_{s\text{-}old})$

- If you are using the “Single run” option, run a program with a D - Diagnostic Step in order to get a new zero offset reading and compensation.
- Run a new test series in order to confirm the correction.
- If a torque scatter test is required (Standard deviation, Cp, Cpk, Cmk, CAM……) a minimum of 20 tightenings are needed.

Double torque transducer

Spindles are calibrated as follows:

Read the result from both transducers in the “Single run” window. The monitoring transducer is the one that the torque results to the reporter(s) come from.

The monitoring transducer is the one that the torque results to the reporter(s) come from.

The control transducer is calibrated in the same way as a single transducer (see Single torque transducer above).

The monitoring transducer can be calibrated at the same time by adjusting the scale factor for transducer 2.

4.17.3.3 Angle

The angle measurement is performed with the resolver signal and is normally not necessary to calibrate (see “wind up” below) as all this is hardware.

The most reliable results are achieved with an inline Torque/Angle transducer and a Torque/Angle amplifier with trace capability, for example an Atlas Copco IRTT transducer with ACTA.

There are two methods of checking the angle measuring.

1. Rotating the spindle in the air. Program a set angle value in the “Test Bolts”-“Single run” menu. Connect your inline transducer and run. Compare the inline transducer angle result with the PowerMACS one.
2. Running on a joint. With this method, a check of the tightening angle is difficult but not impossible to make in a controlled way. Make a tightening on a joint and compare the trace curves, Torque vs. angle, from the two systems.

Results from tests made with an external Torque/Angle recording unit where the angle from a torque level to final torque is presented and then compared with the from PowerMACS reported value are normally **not** giving an acceptable result.

The main reasons for the discrepancy in measured angles are:

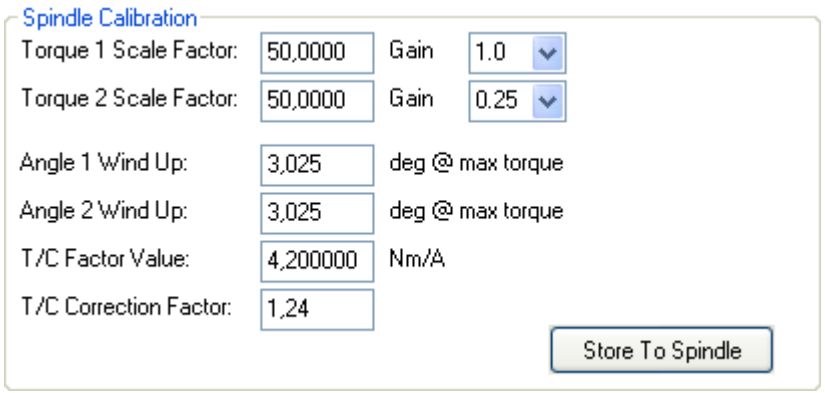
- The start point (torque level) does not occur simultaneously in the control and the monitoring systems
- The stop point is not defined in the same way in the control and the monitoring systems.

4.17.3.4 Wind Up Coefficient

There is a spindle parameter, named **Wind Up Coefficient** (see Spindle Set Up, Calibration page), which is used to compensate for the torsion in the mechanical parts (gears and shafts) between the electric motor shaft and the square drive. The coefficient can be modified in order to fine tune systems where external mechanics (like long thin extension shafts) affects the angle control.

The default value of the **Wind Up Coefficient** is 3 degrees at max spindle torque (that is, the value of parameter **Max Torque**).

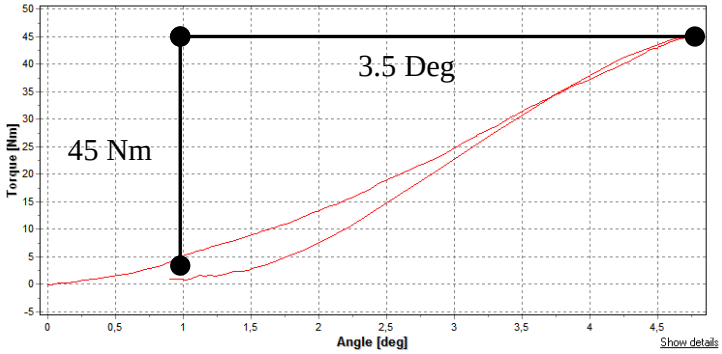
Note! The default Wind Up Coefficient is normally good enough and is in most cases not necessary to change.



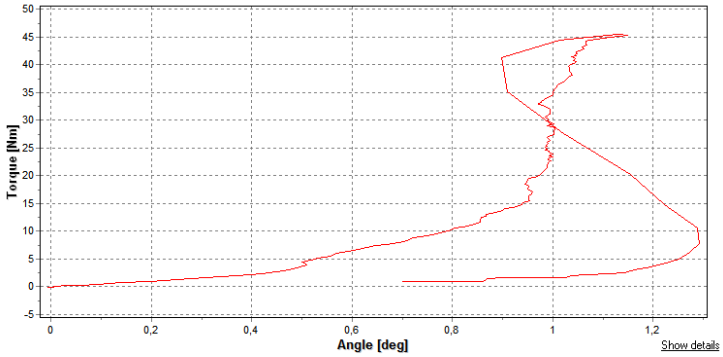
Calibration of the Wind Up Coefficient

It is fairly simple to check and to calibrate the Wind Up Coefficient.

- Create a simple tightening program that has a Diagnostic step, a Torque step and a Cycle End. Program a “high” torque in the T-step, the speed 5 rpm and the Speed Ramp 100 rpm/s. The “high” torque shall be as much as the screw joint can stand without destroying it and that the spindle can give.
- Set the **Wind Up Coefficient** to the value 0.000 deg.
- Run a number of times on the joint without loosening it between in order to have a test joint that is not moving anymore.
- Look at the trace showing Torque vs. Angle. Estimate the angle between two torque levels and calculate the new Wind Up Coefficient. See the example below. The new value of the **Wind Up Coefficient** will be: 3.5 / 45 * 50 (Spindle Max Torque) = 3,88 deg/Nm



- Program the new Wind Up Coefficient in the spindle interface and Apply.
- Run a new tightening and look at the trace again. Now, the curve should look something like the example shown below. As you can see now is there a lash in the socket of about 0.8 deg and then the torque increases to 45 Nm without moving the bolt and this is the goal.



Note! When external equipment is connected on the spindle WindUp is likely to be calculated - but the values should be entered using the External Equipment parameters, see Application External Equipment on the Spindle Set Up form.

4.17.3.5 Current (torque)

The step types TC - Run until Torque with Current Control, DT - Run until Dyna-TorkTM, DT2 - Run until Dyna-TorkTM, method 2, and DT3 - Run until Dyna-TorkTM, method 3 are uses the servo current measurement for step control. The programmed set torque value is converted to a current set value with the T/C factor. The current set value is used in the servo and the servo shuts off itself.

The T/C factor

The spindle parameter **T/C factor** (Torque/Current) (see Spindle Set Up, Calibration page) is the scale factor for all current control. The unit of the **T/C factor** is [<torque_unit>/A] and a theoretical value based on the motor data is used as default setting. The value of the T/C Factor can not be changed since it is written at spindle production and depends on the motor. Instead a **T/C Correction Factor** can be specified to calibrate the torque/current conversions.

One way to calibrate the **T/C Correction Factor** is to select “Current” as control parameter under “Torque usage” in the Spindle Set Up form (see General).

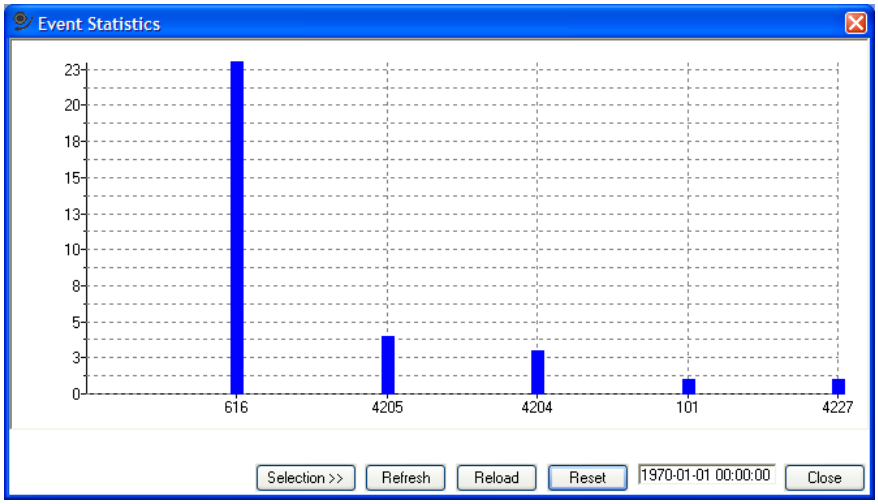
Use the “Single Run” function located of the Test Bolts form.

Run 10 cycles and register the torque values from an inline torque transducer. Calculate a mean torque value and adjust the **T/C-factor** if necessary. Increased TC-factor decreases the true torque.

4.17.4 Event Statistics

The Event Statistics form, invoked using the menu choice **Maintenance** - **Event Statistics…**, can give you a good view on which events that are most frequent.

It displays a so-called Pareto diagram in which the most frequent event type is displayed to the far left. In the diagram you can see both the percentage and the number of times each event has occurred.



One bar is displayed for each event code of which there is at least one error. For a list of possible event codes see chapter: List of events.

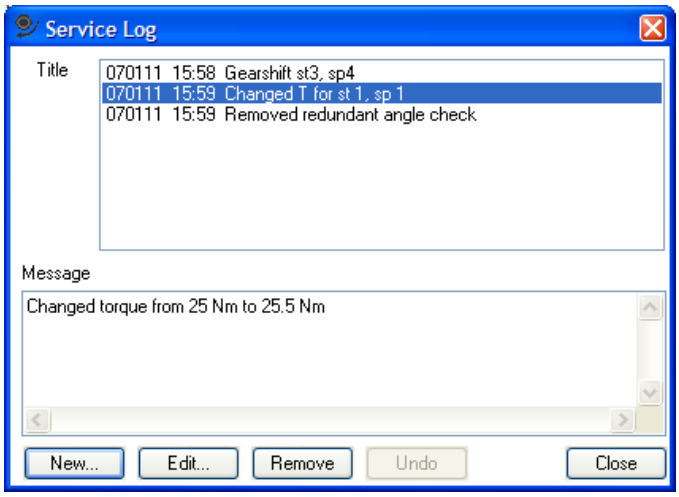
The event statistics are stored in the system in accumulators. You can reset these by pressing the **Reset** button. The last time this was done is shown to the right of this button.

Press the **Reload** button to first clear the accumulators and then reloaded them with all events currently stored in the Event Log (see View Event Log).

To study only certain event types press the **Selection** button to expand the **Event Types frame**. Check each event type you want to include in the display.

4.17.5 Service Log

The Service Log is a useful tool to keep track of changes made to a system. It is invoked from the menu **Maintenance - Service Log…**



Basically it functions as a notepad where you easily can write small notes. Use it to describe all the changes you do to the system, and perhaps more importantly, why you have done them.

The log can contain up to 100 messages and is a part of your setup and will therefore be stored both on disk and in the target system.

When you open the Service Log the titles of all existing logs are listed in the top list box labeled **Title** while the text box labeled **Message** displays contents of the currently selected title.

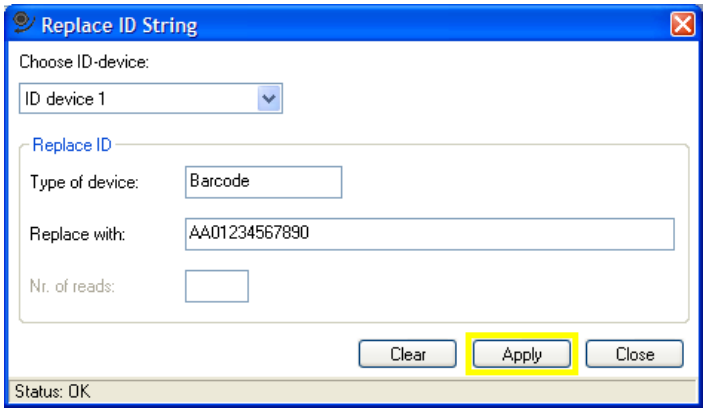
To create a new log entry press **New**. To change an existing message select its title and then press **Edit**. In both cases a form is displayed in which you can enter a proper title and message texts.

To remove a log entry: select its title and press **Remove**.

If you have made changes to the system you will automatically be asked to enter a message in the Service Log when you log out, closes the current setup, or exit ToolsTalk PowerMACS.

4.17.6 Replace ID String

Using the Replace ID String form, opened from the menu **Maintenance - Replace ID String…**, you can manually enter an ID string instead of actually input it using one of the ID-device devices in your system. This is useful if when a barcode label is damaged, or when the ID device for some reason is not working.



The form is accessible only when on-line.

First select the device to operate on using the **Choose ID device** combo box. It is filled with all ID devices defined in the system.

Depending on the type of the selected device the function is as follows:

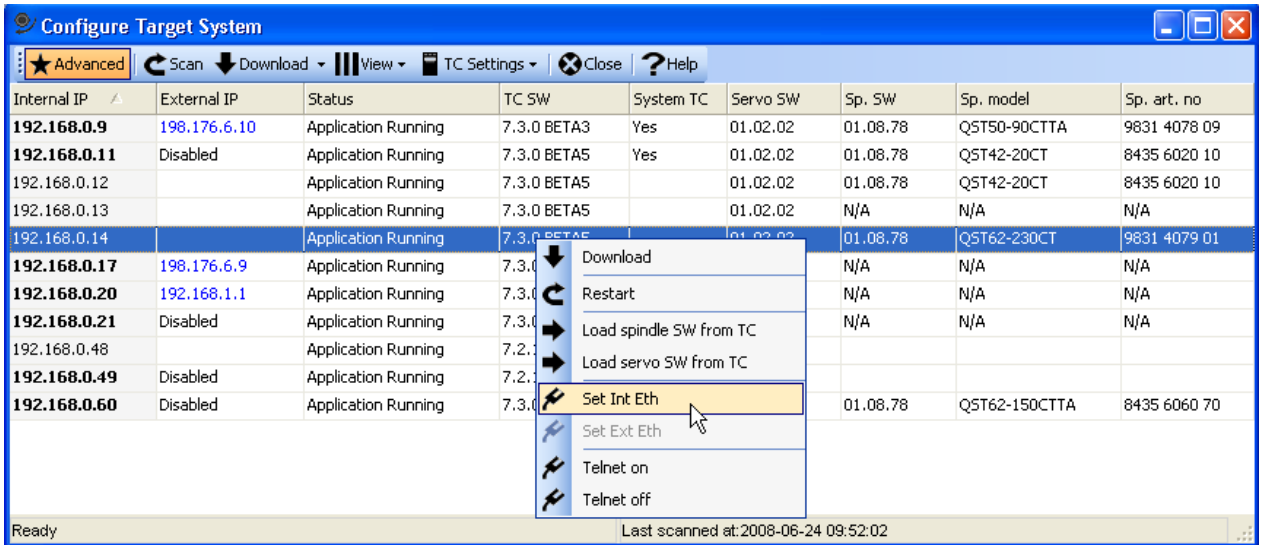
- For type **Barcode**: Enter the new string in the **Replace with** field and press the **Apply** button. The new string is treated by the system just as if it was scanned using the barcode scanner. This means that it is automatically sent to the Station and its PLC (visible as CURRIDSTRING, see Station variables) and is stored in the device.
- For type **Escort AB**, **Escort P&F** or **Omron**: Enter the new string in the **Replace with** edit field and specify how many times the string should be valid in the **No. of reads** edit field. Press **Apply** to send the string to the escort memory device. The next time the PLC orders reading of the device (by generating a positive edge on the PLC ID device variables IDREAD, or one of the Multiple identifier variables) IDTYPE_x_RD, the replacement string will be returned to the Station and PLC instead of actually trying to read data from the escort memory. This will be repeated as many times as set by the **No. of reads** parameter.

Any manually entered ID string sent to the device (by pressing **Apply**) can be cleared by pressing the **Clear** button.

4.17.7 Configure Target System

Selecting the **Maintenance - Configure Target System** menu item opens the Configure Target System form.

This form is used for configuration of your PowerMACS 4000 TCs. This includes functions for downloading new software and changing their IP-address data.



Internal IP	External IP	Status	TC SW	System TC	Servo SW	Sp. SW	Sp. model	Sp. art. no
192.168.0.9	198.176.6.10	Application Running	7.3.0 BETA3	Yes	01.02.02	01.08.78	QST50-90CTTA	9831 4078 09
192.168.0.11	Disabled	Application Running	7.3.0 BETA5	Yes	01.02.02	01.08.78	QST42-20CT	8435 6020 10
192.168.0.12		Application Running	7.3.0 BETA5		01.02.02	01.08.78	QST42-20CT	8435 6020 10
192.168.0.13		Application Running	7.3.0 BETA5		01.02.02	N/A	N/A	N/A
192.168.0.14		Application Running	7.3.0 BETA5			01.08.78	QST62-230CT	9831 4079 01
192.168.0.17	198.176.6.9	Application Running	7.3.0			N/A	N/A	N/A
192.168.0.20	192.168.1.1	Application Running	7.3.0			N/A	N/A	N/A
192.168.0.21	Disabled	Application Running	7.3.0			N/A	N/A	N/A
192.168.0.48		Application Running	7.2.1					
192.168.0.49	Disabled	Application Running	7.2.1					
192.168.0.60	Disabled	Application Running	7.3.0			01.08.78	QST62-150CTTA	8435 6060 70

Press the **Scan** button to bring up a list of all available Tightening Controllers connected to the same physically network as ToolsTalk PowerMACS. The scanning procedure uses network broadcast since this gives the best possibility to find all connected TCs regardless of their IP-address and net mask settings.

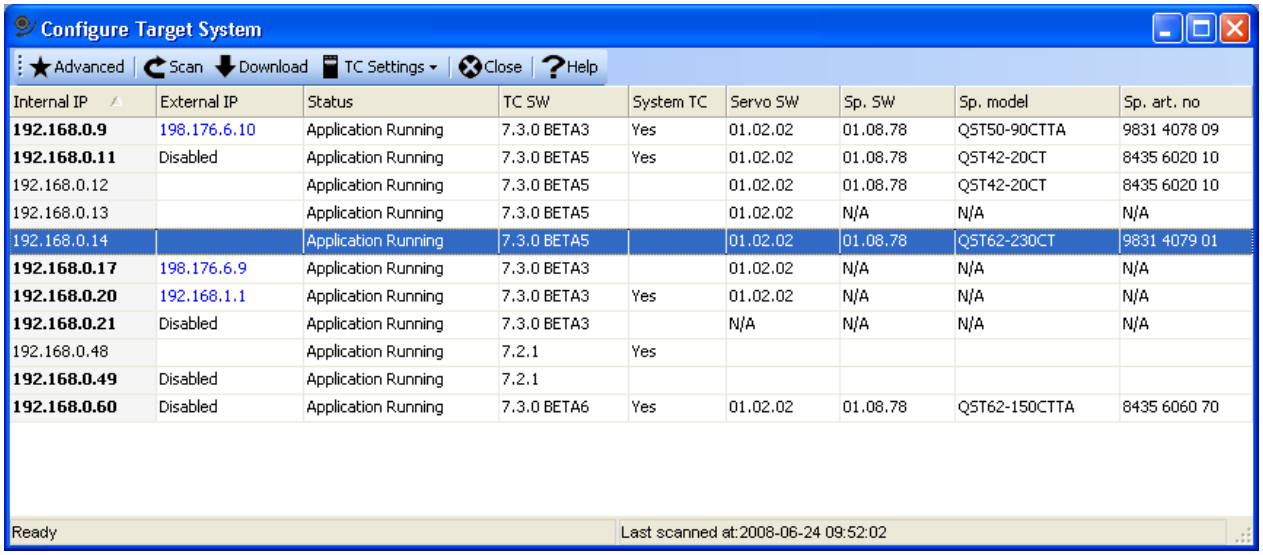
The scan takes approximately 10 seconds and results in a list of TCs that where reached by ToolsTalk PowerMACS. For each TC the following information is always displayed:

- The current **IP address** on both interfaces
- The current **Status** of the TC boot loader. Can be one of the following:
  - **No application** – Means that the boot loader (the very first code executed by a TC when it is powered on) did not find a valid TC System Software when it started. This indicates that the System Software is either missing or is corrupt. To correct this try to download System Software to the TC.
  - **Application Running** – Means that the boot loader did find a valid System Software which it started. It does not indicate that the TC has a setup loaded.
  - **Waiting for download** – Means that the TC is set up to start download of new System Software when it is restarted.
  - **Downloading: nn %** – Indicates that download of System Software is progress.

- **Download Ok** – Means that the boot loader has successfully downloaded System Software to the TC. The TC must be restarted for the new software to take effect.
- **Download failed** or **Download Aborted** – Means that the boot loader failed to download System Software to the TC. The cause of this may be network problems. Try to download again.

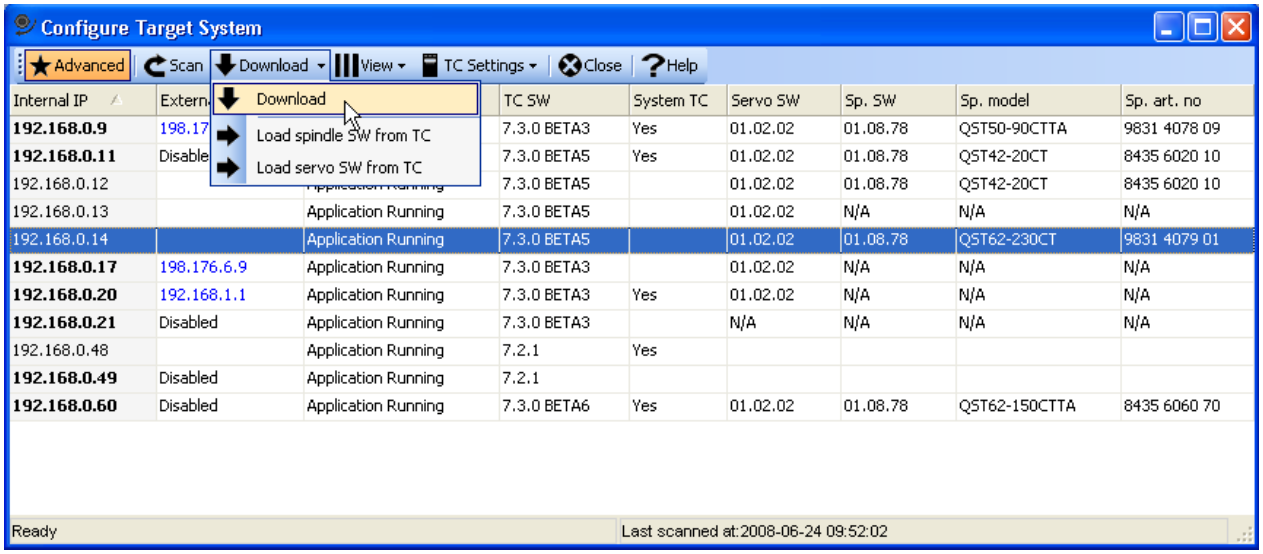
4.17.7.1 Advanced mode

When not in advanced mode, the form shows only the most basic functions.



Advanced mode is reached by clicking the **Advanced** button on the toolbar. Advanced mode makes the following available:

- **Restart** button
- Changes **Load TC SW** from a button to a button with a menu. The menu contains what was earlier the **Manage software** function, to load spindle and servo SW.
- **View** menu, where much more information to show can be chosen.
- Context menu when right-clicking on either column headers or on a TC in the list.
- Possibility to turn on/off telnet.



4.17.7.2 Restart of TC's after download

When all selected TC's have successfully downloaded new software, the user is promted to restart them.

4.17.7.3 Information fields

The information fields are divided into groups **Default**, **Spindle SW**, **Spindle Info**, **Servo**, **TC SW**, **TC HW**, and **IP Addresses**. When klicking on a group in the **View** menu, the list is refreshed to only show the fields from the selected groups. The groups contain the following info fields:

Default

- TC SW
- System TC
- Servo SW
- Sp. SW
- Sp. model
- Sp. art. no

Spindle SW

- Sp. model
- Sp. SW
- Sp. SW in TC
- Sp. boot

Spindle HW

- Sp. model
- Sp. serial no
- Sp. art. no
- Sp. HW serial no
- Sp. HW ver
- Sp. cycles
- Sp. cycles since service
- Sp. cycles to next service

Servo

- Servo SW
- Servo SW in TC
- Servo boot
- Servo HW ver
- Servo serial no

TC SW

- TC SW
- TC boot
- Telnet

IP addresses

- Internal net mask
- Gateway
- Internal MAC
- External net mask
- External MAC
- Com status

TC HW

- TC model
- TC serial no

- TC art. no
- PCB serial no
- PCB type
- PCB ver.
- Fieldbus

If the TC is configured as a **System TC**. The System TC is the TC that controls all other TCs included in a PowerMACS system. It is it that ToolsTalk PowerMACS is connected to and that holds the setup of the system

TC boot is the version of the TC Boot loader. "(B1)" or "(B2)" indicates which of the dual boot loaders, "Boot 1" or "Boot 2" that where active during start up. See chapter: Prepare TC for download of software for a description of the dual boot loaders.

For a System TC, more data may be displayed if clicking **Additional Info** in the **View** menu. This additional data includes the name of the System, the number of stations, the total number of bolts, and the number of TCs used by the system. It is always the actually used values of the parameters that are displayed in the list, this is important to understand when changing net data parameters.

Note! Since the scan procedure uses broadcasts it will detect all TCs that are connected to your network regardless of their IP address configuration.

However, this does not mean that you can connect to, or send all configuration commands, to all TCs displayed in the list. The reason for this is that the normal communication between ToolsTalk PowerMACS and a TC is done using TCP and UDP which requires the communicating nodes to be on the same IP network, or that a route between the networks is known. Therefore whether or not you can connect to a TC depends on its IP-address, net mask, and default gateway.

To help you detect this situation an UDP message is sent to each TC found after the scan procedure. If the TC does not bounce this message back the node is marked as erroneous and the cause is displayed in the field under list, or in the **Com status** field.

Changing IP-address configuration, preparation of System Software download and changing service data settings may be done to multiple TCs at the same time. Select the TCs of interest by marking them in the list. To select a range click on the first row, hold down the Shift key and click on the last. To select multiple single items, hold down the Control key and click on the wanted items.

To modify the network settings press the **Set Int Eth** or **Set Ext Eth** button. This will open the

Change Net Data form.

Press the **Download** button to prepare download of System Software using the form Prepare TC for download of software.

To enable the telnet interface on a TC klick the **Telnet** menu item.

Note The Telnet server is only used for advanced debugging purposes and **should normally be switched off** during normal operations. **You should not activate it unless directed by Atlas Copco personal to do so**.

The server is automatically disabled whenever the TC is restarted.

This command requires that the Console Computer is on the same network as the target system(s) to modify, or there must exist a route between them.

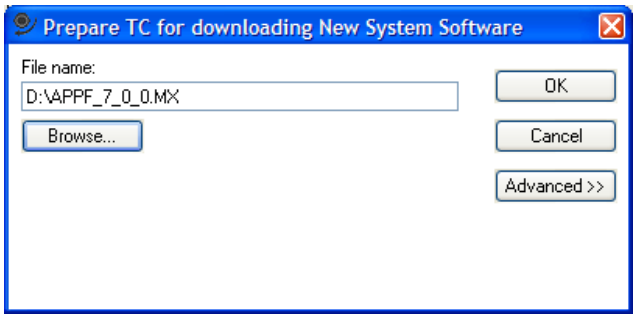
For new net data, or a download to be effective, the TC must be restarted. Turning the power off and on again is the normal way to do this. But if at least one TC accepts the request you will be offered the possibility to restart those TCs remotely. You should only restart the TCs remotely if you are absolutely sure that it is safe for the system to do so.

To overwrite spindle software with spindle software stored in TC flash, klick **Load spindle SW from TC**. This is only needed when wanting to downgrade the software in the spindle. TC automatically upgrades the spindle upon start of communication if the spindle software in TC is newer than the software in the spindle.

To overwrite servo software with servo software stored in TC flash, klick **Load servo SW from TC**. This is only needed when wanting to downgrade the software in the servo. TC automatically upgrades the servo upon start of TC if the servo software in TC is newer than the software in the servo.

4.17.7.4 Prepare TC for download of software

This form is used to select which software file to load to TCs selected for the operation.



To reduce the risk for non-repairable errors caused by failures during software download the application is divided in two parts, the **boot loader** and the **application**.

The boot loader contains only the functions needed for starting and upgrading the application part while the application contains the tightening functionality etc. Normally you need only to update the application.

Therefore the TC Software module is always delivered in two files, named as follows:

- `BOOT_x_y_z.MX`: This file contains the boot loader image
- `APP_x_y_z.MX`: This file contains the application image

x_y_z above is the version number of the respective file. Example: "`APP_7_0_0.MX`", `BOOT_2_1_7.MX`", etc.

The boot loader is duplicated to increase the safety when updating the boot loader part. This gives each TC dual boot loaders, named "Boot 1" and "Boot 2". At start up Boot 1 is responsible for checking that Boot 2 is OK and if so start Boot 2. Boot 2 then performs the "everyday" work of the boot loader, that is, starts the application, download new code etc. Should Boot 2 be found incorrect Boot 1 would instead handle the work of the boot loader.

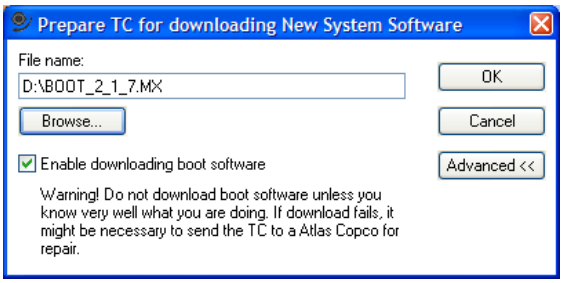
The boot loader file BOOT_x_y_z.MX contains both Boot 1 and Boot 2. If Boot 1 exists in the TC it is not replaced when such a file is loaded since it is used only as a backup in case Boot 2 would be found incorrect.

Note1 You should never downgrade the boot loader software. The latest boot loader version is compatible with ALL applications.

Follow these steps to download new software:

1. Press **Browse…** to locate the file to load.

   If boot must be downloaded, click on the **Advanced >>** button and check the **Enable downloading boot software** check box that is displayed. Select the file named `BOOT_x_y_z.MX`. Please note the warning.



2. Click **OK** to **prepare** the selected TCs to download the specified file. This will cause a "Prepare TC for download" command to be sent to the TCs. The success or failure of this command is displayed in the Configure Target System dialog.
3. To start the actual download, you must restart the TCs. When restarted, each TC will request the new file from ToolsTalk PowerMACS.
4. During download, TCs will indicate that they are downloading by flashing a text on the display.
5. When the download is finished the result of the download is displayed both by the TCs and ToolsTalk PowerMACS. If successful the TC displays “**Download OK**” and if it failed “**Download Failed**”. See chapter: Configure Target System for how ToolsTalk PowerMACS indicates the result.

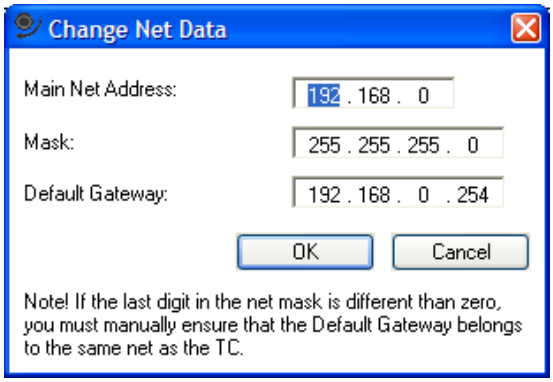
Note! It is important that you do not interrupt the download process when started. If that happens by accident while downloading **boot software** you should **not restart** the TC. Restarting it may cause the boot code to be erased, which will prohibit any further download. Instead try to download the software again.

Restart the TCs after download to make the new software effective.

If you experience problems with spontaneously aborted download, it could be your network card that has a cable hold off timeout set. For Windows 2000, look in Device Manager (open Control Panel, select System, select tab Hardware and click on Device Manager). Find your network card under the heading Network adapters and double click on it to view Properties. Select the tab Advanced and look for the Property "Cable hold off timer" or similar. If this property has a value greater than zero, you could experience the problem mentioned above. Set this property to zero. Note! Not all network cards has this property.

4.17.7.5 Change Net Data

This form is used for changing the network settings for one or more TCs.



Enter the new values in the field displayed

The **Main Net Address** determines the first three numbers in the IP address for the TC. The TC number set on on the TC display gives the last number of the address.

If you set a network mask with a nonzero value as the last number you must manually ensure that the **Default Gateway** belongs to same network as all the TCs.

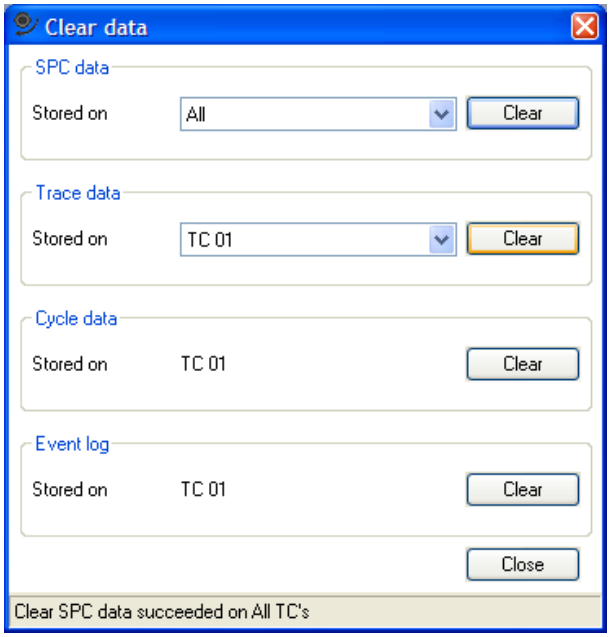
Click **OK** to send the new values to the TCs. The result of this request is displayed in the Configure Target System form (see chapter: Configure Target System).

Keep in mind that the new values will not be displayed in the list until the TCs are restarted. The reason for this is that the list always shows the currently used data and the TC will not use the new values until it is restarted.

Note! This command requires that the Console Computer is on the same network as the target system(s) to modify, or there must exist a route between them.

4.17.8 Clear data

The Clear data form is invoked using the menu item **Maintenance – Clear data…**.



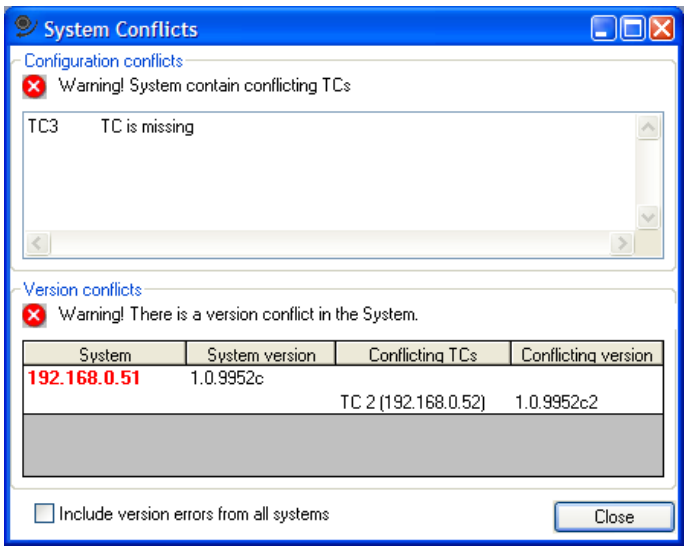
The Clear Data form is accessible only when on-line and is used to clear process data stored in the non volatile memory on the TCs. This includes the following:

- SPC data, on all TCs or on a individual TC. The SPC data for a bolt is stored on the TC that controls the spindle used to tighten the bolt.
- Trace data, on all TCs or on a individual TC. The Trace data for a given bolt is stored on the TC that controls the spindle used to tighten the bolt.
- The Cycle data of the system. Only stored on the System TC, that is, the first TC in the system.
- The Event log of the system. Only stored on the System TC.

To clear any of the above listen items first select the wanted TC, or TCs, and then press the **Clear** button.

4.17.9 Check for System Conflicts

The Check for System Conflicts form, normally invoked using menu item **Maintenance – Check for System Conflicts…**, is a tool for checking if the system contains conflicting TCs and/or TCs with different versions of the TC System Software.



The frame **Configuration conflicts** lists all Spindle TCs that the System TC cannot connect properly to. The list contains the Spindle TCs number with in the system (TC 1 is the System TC) and the reason for the problem. This information is useful when trying to locate overlapping system layouts, that is, several System TCs trying to use the same Spindle TC. The same information can also be found in the System Map.

The data in the **Configuration conflicts** frame covers only the system that ToolsTalk PowerMACS is currently connected to, and only when connected.

ToolsTalk PowerMACS continuously listens for telegrams (multicast) sent from a TC, System TC or Spindle TC, that has detected a version conflict, that is, a System TC and a Spindle TC that does not use the same version of the System Software. All errors of this type are presented in the **Version conflicts** frame. This function is active also when ToolsTalk PowerMACS is not connected to any target system. However, when on-line against a particular system the information received directly from it is used to update the form with less delay.

Check the **Include version errors from all systems** to show version problems reported also for TCs belonging to other systems then the currently selected target system.

Please note that the TCs do broadcasts the version conflict information with rather low frequency. This leads to that the information in the Version conflicts frame may display an error even though it just has been corrected. The information shown can be up to 30 seconds old.

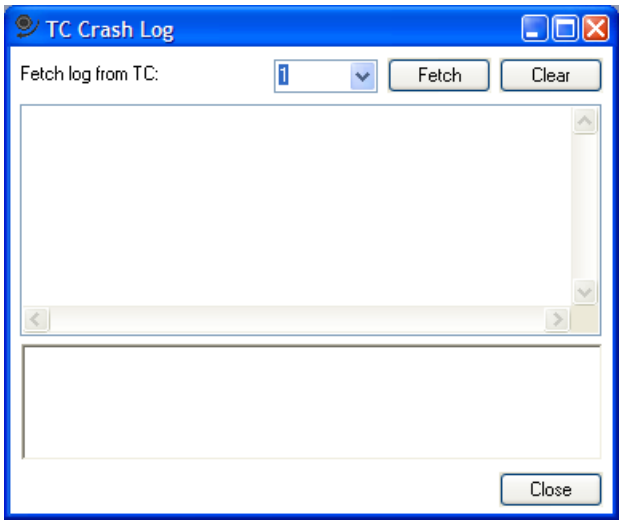
If any of the above conflicts are detected for the selected system when going on-line then a warning is written on the caption bar of ToolsTalk PowerMACS and this form is displayed automatically. It is not possible to close the form as long as there are errors in the system.

See also chapter: Tightening Controller for important information on System and Spindle TCs and The TC HMI Menu System for how versions errors are displayed directly on the TCs. Chapter: Configure Target System describes for how to download new TC System Software (Prepare TC for download of software) and how to configure a TC as a System or a Spindle TC (The TC HMI Menu System).

4.17.10 TC Crash Log

The TC Crash Log form is invoked using menu item **Maintenance – TC Crash Log…**, is a tool that is used to collect debug information from one or more TCs that crashed due to an internal software error.

Normally you should not run in to situations where this tool is needed but if you do the data it retrieves can be of great help to Atlas Copco when analyzing the fault.

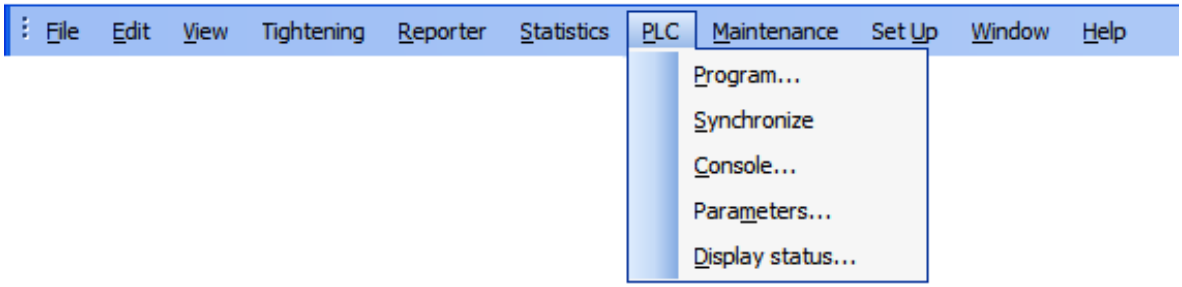


Use the **Fetch log from TC** combo box to select which TC to upload data from. Choosing the **All** alternative will make all TCs be polled for data.

5 PLC

5.1 PLC - Overview

In every PowerMACS station there is a PLC (Programmable Logical Controller), called PowerMACS PLC. This is programmed in a standardized, graphical, way according to the standard IEC 61131-3.



This part describes how to set up and use the PLC. A general description of the PLCs role in, and interface to, the PowerMACS system is given in the next two chapters: General and System Globals.

The basics on how to write a program are given in the Programming the PLC chapter while a small, but still powerful, user interface for the PLC is described in the PLC Console chapter.

Thorough information on how to use the Multiprog wt environment for editing PLC programs is given in "The PowerMACS PLC Manual [2].

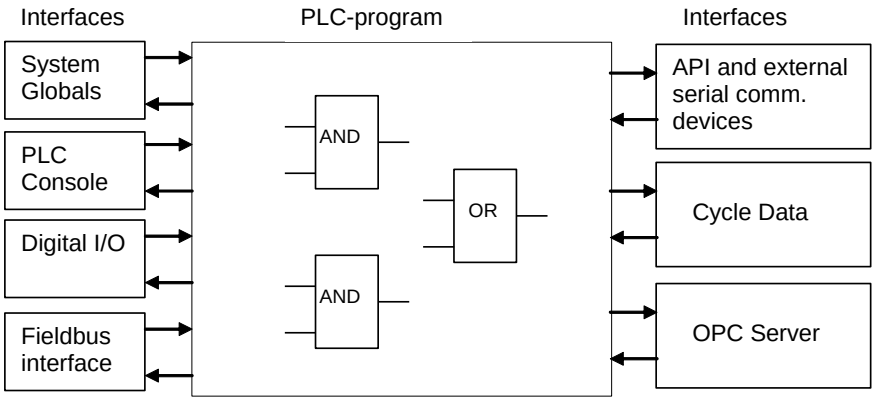
How to access parameters inside the PLC using the ToolsTalk PowerMACS is described in chapter: PLC Parameters.

5.2 General

PowerMACS PLC has a primary assignment to interface customer control system and peripheral devices through different communication interfaces. It is used as ”glue” between various functions within the system, as for example:

- Start of tightening from digital inputs
- Output of status to digital outputs
- Controlling flow of data
- Analyzing ID codes

The PLC has interfaces to several other units in the PowerMACS system.



The *System Globals* are inputs that represent the state of, and outputs that are used to control the tightening process within PowerMACS. All these signals are described in chapter: System Globals.

The *PLC Console* interface is a simple, but still powerful, interface between the PLC program and the ToolsTalk PowerMACS application running on the Console Computer. See chapter: PLC Console for a detailed description of it.

From *Digital I/O* the PLC can read digital inputs (push buttons, switches etc.) and write digital outputs (lamps, relays etc.) that have been configured in PowerMACS using an I/O device. See chapter: I/O Device for how this is done.

The *Fieldbus interface* makes it possible to access signals in the PowerMACS PLC from a fieldbus master. From the master it is then possible to write and read data in order to control the system and check its status. See chapter: Fieldbus Interface for a description of this interfaces and how it is configured.

The *API and external serial communication* interface makes it possible to access signals in PLC from external application from an external PC program or via some serial communication protocols. This is handled much in the same way as the fieldbus interface. See chapter: API, Application Programmers Interface and Serial Communication for a description of these interfaces and how they are configured.

The *Cycle Data* interface makes it possible to read data produced as cycle data result. See the description of PLC in chapter: Peripheral Devices, for how to enable this function.

5.3 System Globals

The *System Globals* is the interface between the PLC and the tightening controller part of the system. By setting global outputs from the PLC you can order ID devices to be read, start a tightening cycle and so on. By reading inputs you can check the state of the system, for example if a cycle is ready.

These variables are all declared in the **System_Globals** section of the Global Variables worksheet found in the project tree of the PowerMACS PLC.

5.3.1 Station variables

The following inputs reflects the status of the station, and to some extent, the system:

Variable	Type	Use
EMERGSTOPIN	BOOL	This signal is True if the TC hardware Emergency input is inactive.
CURRSYNCH	INT	Reflects the number of synchronization points that have been passed. Initialized to 0 at cycle start.
CYCLEDATASTORED	BOOL	This input is set False when START is set True and is set to TRUE when the Cycle Data result has been stored in the PowerMACS internal cycle data queue. That is, when True an external device can read the cycle data without delay. This might be useful for external devices that need to know when they can expect the result of the last cycle is available.
SYSTEMSTATUS	INT	Indicates system errors. The respective bits have the following meaning (bit 0 is the least significant): • 0: (value 1) Servo errors • 1-31: Reserved
TESTBOLT_ACTIVE	BOOL	This signal is True when the ToolsTalk PowerMACS Test Bolt form is open and connected to this station. See also chapter: Test Bolts.
ACTA_ACTIVE	BOOL	This signal is True when an ACTA 3000 unit is connected to the system and is active against a spindle belonging to this station.
TTPM_ACTIVE	BOOL	Indicates whether or not a ToolsTalk PowerMACS is connected to the target system. True if a connection exists and False if not.
VALID_MODE	BOOL	This signal is True when a valid the mode is set using the PLC output MODE. A mode is valid if it is in the interval [1...50] and at least one bolt is connected normally for the mode.
ACTA_MODE	INT	Reflects the mode number sent from the ACTA 3000 unit. Only valid if ACTA_ACTIVE is True.
ACTA_SPINDLENO	INT	Reflects the spindle number sent from the ACTA 3000 unit. Only valid if ACTA_ACTIVE is True.

Variable	Type	Use
CD_OVERRUN_W1	BOOL	This input is set True when there is room for less than 10 more cycle data in the cycle data queue for any device. If set, it is automatically reset when all devices have space for at least 10 more data, that is when the failing device has read its queue
CD_OVERRUN_W2	BOOL	Same function as CD_OVERRUN_W1 with the difference that it indicates that at least one queue is full but not yet overrun. That is, the next cycle data will be lost.
CD_OVERRUN_E1	BOOL	This input is set True when a new a cycle data caused the queue for at least one device to be overrun. If set, it remains set until a new cycle can be added to the queue without causing it to be overrun. This means the flag is not reset immediately when the failing device read a cycle data from the queue (which frees one position in the queue). The status is not checked with less than that a new cycle data is generated.
CURRMODE	INT	This input always reflects the MODE value currently known by the station.
EVENT	BOOL	System events. Is True when there are unobserved events of any type (see View Event Log).
SPCEVENT	BOOL	This input is True when there is unobserved "SPC" events in the event log (see View Event Log).
ACCESSEVENT	BOOL	This input is True when there is unobserved "Access" events in the event log (see View Event Log).
POWERUPEVENT	BOOL	This input is True when there is unobserved "Power up" events in the event log (see View Event Log).
CHKEVENT	BOOL	This input is True when there is unobserved "Check" events in the event log (see View Event Log).
SWEVENT	BOOL	This input is True when there is unobserved "Software" events in the event log (see View Event Log).
SYSERREVENT	BOOL	This input is True when there is unobserved "System error" events in the event log (see View Event Log).
MODEVENT	BOOL	This input is True when there is unobserved "Modification" events in the event log (see View Event Log).
HWEVENT	BOOL	This input is True when there is unobserved "Hardware" events in the event log (see View Event Log).
EMERGEVENT	BOOL	This input is True when there is unobserved "Emergency stop" events in the event log (see View Event Log).
GENERALEVENT	BOOL	This input is True when there is unobserved "General" events in the event log (see View Event Log).
XCOMEVENT	BOOL	This input is True when there is unobserved "External communication" events in the event log (see View Event Log).
SETUPEVENT	BOOL	This input is True when there is unobserved "Setup" events in the event log (see View Event Log).

Variable	Type	Use
CONS_IN_1	BOOL	Input connected to the ToolsTalk PowerMACS form PLC Console.
CONS_IN_2	BOOL	Input connected to the ToolsTalk PowerMACS form PLC Console.
CONS_IN_3	BOOL	Input connected to the ToolsTalk PowerMACS form PLC Console.
CONS_IN_4	STRING(40)	Input connected to the ToolsTalk PowerMACS form PLC Console.
CONS_IN_5	STRING(40)	Input connected to the ToolsTalk PowerMACS form PLC Console.
SERVOSTATUS	ARRAY[1..50] OF BYTE	Reports the current status of each servo (TC) in the system. The respective bits have the following meaning (bit 0 is the least significant): • 0: (value 1) Servo NOK • 1: (value 2) Communication NOK • 2: (value 4) Over temperature • 3: (value 8) Spindle not ready • 4: (value 16) Reserved • 5: (value 32) Reserved • 6: (value 64) Reserved • 7: (value 128) Any Fault See also the description of the PLC input RESET.
SPINDLESTATUS	ARRAY [1..50] OF BYTE	Reports the current status of each spindle (connected to TC) in the system. The respective bits have the following meaning (bit 0 is the least significant): • 0: (value 1) Spindle NOK • 1: (value 2) Communication NOK • 2: (value 4) Over temperature • 3: (value 8) Servo not ready • 4: (value 16) Reserved • 5: (value 32) Reserved • 6: (value 64) Reserved • 7: (value 128) Any Fault See also the description of the PLC input RESET.

<table>
<tr><th>Variable</th><th>Type</th><th>Use</th></tr>
<tr><td>DEVICE_CTRL</td><td>ARRAY [0..31] OF {DATAHOLD, DATADROP}</td><td>Controls device queue
<table>
<tr><th>DataHold</th><th>DataDrop</th><th>Result</th></tr>
<tr><td>False</td><td>False</td><td>Cycle data is saved for this device.</td></tr>
<tr><td>False</td><td>True</td><td>Cycle data is not saved for this device.</td></tr>
<tr><td>True</td><td>X</td><td>Cycle data is saved for this device but the wanted and undread bits are not set (datahold bit is set instead).</td></tr>
<tr><td>True</td><td>False
True</td><td>Cycle data is dropped for this device when a positive edge is detected on datadrop (latest or specific ID).</td></tr>
<tr><td>True
False</td><td>X</td><td>Cycle data is kept for this device when a negative edge is detected on datahold (cycle data that have the datahold bit set is marked as wanted and unread).</td></tr>
</table></td></tr>
<tr><td>STEP_OK</td><td>BOOL</td><td>Set when all bolts in the cycle have reached a synchronization point and have status OK. It will stay set until next step start. The minimum time is one PLC cycle.</td></tr>
<tr><td>STEP_NOK</td><td>BOOL</td><td>Set when all bolts in the cycle have reached a synchronization point and at least one bolt has status NOK. It will stay set until next step start. The minimum time is one PLC cycle.</td></tr>
<tr><td>PLC_DI_1..10</td><td>BOOL</td><td>These signals can be set at specific times directly from the tightening programs, e.g. at step start or zone end. See Ramps & Other – Signals at step start and end (only available for Gauging)</td></tr>
</table>

Outputs used for controlling the station:

Variable	Type	Use
MODE	INT	Set to number of mode to use in next cycle. Sampled on the positive edge of START.
RESET	BOOL	Reset servo errors on a positive edge. Should only be set True if the station is idle. **Note!** All servos connected to the station are reset.
START	BOOL	Starts cycle. A positive edge will start the first step in the cycle. This signal is ignored if the station is not idle, or if STEPSTOP, EMERGSTOP or MACHINESTOP is True.
LOOSENING	BOOL	Loosen cycle. When set to True mode 50 will be started and all cycle data and trace data will be automaticly dropped. See detailed description below. Note that the cycle and trace data that is produced while loosening is active is dropped and will not be stored in the system.
STEPSTOP	BOOL	Step stop. Set to True to stop current step with status OK. Next START will continue with next step in the cycle.
EMERGSTOP	BOOL	Emergency stop. Set to True to stop the cycle with status NOK. Next START will start cycle from beginning.
MACHINESTOP	BOOL	Machine stop. Set to True to stop the cycle with status NOK. Next positive edge on START will start a new cycle.
MONSTART	BOOL	Start monitoring. Set to True to start monitoring of the spindles. This signal is ignored if the station is not idle.
MONEND	BOOL	End monitoring. Set to True to end monitoring of the spindles.
DATAHOLD	BOOL	Data hold. Set to True, before starting a cycle, to hold the transmission of cycle data. The data is released when the input is set to False if the station is idle (status < 4) and DATADROP is False. Note that the cycle data that is produced while hold is active will contain only one station header, the one produced by the first start.
DATADROP	BOOL	Data drop. If True when a cycle data is to be reported then the data is dropped, without any distribution. Cycle data is normally reported when the cycle ends but if HOLDDATA is active it is reported when DATAHOLD is set to False.
TRACEDROP	BOOL	Trace drop. If True when the cycle end the traces will not be distributed to any device.
GENEVENT	INT	Set to a code 1-10 to generate an event. The code used will be displayed together with the event.
STEPTYPE	INT	Control data used for the PLC - Run PLC step used in the tightening program. Defines which control function the step should execute. See the step description for valid values.
STEPVAL	REAL	Control data for the PLC - Run PLC step. Used as target value.

Variable	Type	Use
GENOUT_1 ... GENOUT_5	INT	General output. Can be used for selection of double encoders or change of spindle direction. See chapter: Spindle Set Up.
FREESTRING	STRING(40)	The value of this output can be included in the cycle data result as the station level result variable "Free Str". Sampled on the positive edge of START.
FREESTRING2	STRING(40)	The value of this output can be included in the cycle data result as the station level result variable "Free Str 2". Sampled on the positive edge of START.
FREESTRING3	STRING(40)	The value of this output can be included in the cycle data result as the station level result variable "Free Str 3". Sampled on the positive edge of START.
FREENUM1	DINT	The value of this output can be included in the cycle data result as the station level result variable "Free No 1". Sampled on the positive edge of START.
FREENUM2	DINT	The value of this output can be included in the cycle data result as the station level result variable "Free No 2". Sampled on the positive edge of START.
DISABLE_TESTBOLT	BOOL	Set this output TRUE to disable the usage of the ToolsTalk PowerMACS Test Bolt form for this station. See also chapter: Test Bolts.
ACKACCESSEVENT	BOOL	Set this output True to mark all events of type "Access" as observed (see View Event Log and PLC Event handling).
ACKPOWERUPEVENT	BOOL	Set this output True to mark all events of type "Power up" as observed (see View Event Log and PLC Event handling).
ACKCHKEVENT	BOOL	Set this output True to mark all events of type "Check" as observed
ACKSWEVENT	BOOL	Set this output True to mark all events of type "Software" as observed (see View Event Log).
ACKSYSERREVENT	BOOL	Set this output True to mark all events of type "System error" as observed (see View Event Log).
ACKMODEVENT	BOOL	Set this output True to mark all events of type "Modification" as observed (see View Event Log).
ACKHWEVENT	BOOL	Set this output True to mark all events of type "Hardware" as observed (see View Event Log).
ACKEMERGEVENT	BOOL	Set this output True to mark all events of type "Emergency stop" as observed (see View Event Log).
ACKGENERALEVENT	BOOL	Set this output True to mark all events of type "General" as observed (see View Event Log).
ACKXCOMEVENT	BOOL	Set this output True to mark all events of type "External communication" as observed (see View Event Log).
ACKSETUPEVENT	BOOL	Set this output True to mark all events of type "Set up" as observed (see View Event Log).

Variable	Type	Use
ACKSPCEVENT	BOOL	Set this output True to mark all events of type "SPC" as observed (see View Event Log).
CONS_OUT_1 ... CONS_OUT_5	STRING(40)	Outputs connected to the ToolsTalk PowerMACS form PLC Console.
RESETSTATUS	BOOL	Set this output True to reset the status of the station, and all bolts of the station, in both the PLC and ToolsTalk PowerMACS. A positive edge on this output will affect the following: • The PLC inputs STATIONSTATUS_EXT and BOLTSTATUS_EXT[n] all flags are cleared • The status of the station and all bolts displayed by the ToolsTalk PowerMACS forms “System Map” and “Assembly Overview” are cleared by setting their background color to grey and removing all result information being displayed **Note!** A positive edge on RESETSTATUS while the station is not idle is silently ignored.
DISABLE_ACTA	BOOL	Set this output True to disable the activation of an ACTA 3000 unit for any spindle belonging to this station.
FILTER_SPINDLEEV	BOOL	Filter spindle events. When signal is high, spindle and servo events are not generated.
INTERMEDSTART	BOOL	In each step it will be possible to select if the station shall continue with the next step directly at step end or if it shall wait for the PLC to continue. If the checkbox “Wait for PLC at step end” is checked for at least one bolt in the cycle the station will stop and wait for the signal INTERMEDSTART from the PLC to continue running the cycle.
PLC_DO_1..10	BOOL	These signals can be used as input to a JOG - Run until digital input goes high / low or W - Wait step

Note! For the respective ACKxxxEVENT outputs to function they must first be enabled using the Advanced Station Settings form. See chapter: PLC Event handling).

5.3.1.1 Loosening cycles

It is possible to run a loosening cycle with automatic data and trace drop by using the digital I/O signal that has been designated for this purpose (LOOSEN). To use this function use the digital I/O signal named DI_Loosening and mode 50 in the mode table in the following way:

- In the mode table ensure that Mode 50 contains the loosening program that is to be run.
- Activate the digital I/O signal DI_Loosening by setting it high.
- Start the cycle using the digital I/O signal DI_Start.
- Deactivate the digital I/O signal DI_Loosening by setting low.

The loosening functionality is only activated if the tightening cycle is started from the PLC, it does not affect cycles started from TESTBOLT, ACTA or GM-DEVICNET devices.

The standard PLC program that is generated by the wizard contains signals, outputs and functions to support this. It is off course possible to alter the PLC program to use other means or input rather than the digital I/O signal DI_Loosening by using the PLC output variable LOOSENING.

Note! Mode 50 in the mode table is acting as any other mode when the digital I/O signal DI_Loosening is active low.

PLC

The following shared memory variable also reflects the configuration and status of the station and system. Even though they are defined as shared memory variables, which mean that the PLC application can write to them, they should be used as inputs only.

Variable	Type	Use
NO_OF_BOLTS	INT	Indicates the number of bolts defined for this station.
NO_OF_SPINDLES	INT	Indicates the number of spindles (and TCs) defined for this station.
IO_INCLUDED	BOOL	If True an I/O device is configured to be located on the same TC as this station runs on.
FB_INCLUDED	BOOL	If True a Fieldbus device is configured to be located on the same TC as this station runs on.
INT_PARAMS	INT_PARAM_ARR	This is an array of 130 parameters of type INT (16 bit signed integer) defined and modified using the ToolsTalk PowerMACS forms PLC Parameters Set Up and PLC Parameters. They can be used to make the PLC more generic in the sense that minor parameter changes that earlier must be done using the Multiprog wt editor now can be done by the end user using the normal ToolsTalk PowerMACS interface.
REAL_PARAMS	REAL_PARAM_ARR	This is an array of 20 parameters of type REAL (32 bit floating point) defined and modified using the ToolsTalk PowerMACS forms PLC Parameters Set Up and PLC Parameters. See also the description of INT_PARAMS.
STRING_PARAM_1 ... STRING_PARAM_10	STRING40	These parameters of type STRING40 (max 40 character long strings) are defined and modified using the ToolsTalk PowerMACS forms PLC Parameters Set Up and PLC Parameters. See also the description of INT_PARAMS.

Variable	Type	Use
STATIONSTATUS_EXT	STATION_STATUS_EXT_TYPE	This variable indicates the station status of the last tightening. It is a structure containing the following BOOL flags: • **OK** True if result is OK or OK after RM (OKR). • **OKRM** True if result is OK after RM. • **NOK** True if result is NOK or Stop and Terminate Not OK (TERMNOK). • **TERMNOK** True if result is Stop and Terminate Not OK (TERMNOK). The value of this variable is only changed as a result of running a cycle. When the station is started all flags are reset and when the cycle is ready the relevant flags are set. Use this variable together with STATIONSTATE as an alternative to the STATIONSTATUS input.
STATIONSTATE	STATION_STATE_TYPE	This variable indicates the current state of the station. It is a structure containing the following BOOL flags: • **IDLE** True if the station is idle, that is, it is not executing a cycle. • **RUNNING** True if the station is running a cycle, that is, it is not idle. This includes evaluating and waiting for PLC as well. • **EVALUATING** True if the station is evaluating the results from the bolts. • **WAITING_FOR_PLC** True if the station is waiting for a new START signal due to that some bolt is executing a step that requires a this to continue. Use this variable together with STATIONSTATUS_EXT as an alternative to the earlier STATIONSTATUS input.

Variable	Type	Use
SYS_DEVTYPE_ERR	DEVTYPE_ERR_ TYPE	This variable indicates erroneous devices in the system per device type category. The DEVTYPE_ERR_TYPE is a structure containing a BOOL flag for each type of device, plus TC and Servo, that can be included in PowerMACS system: • **ANYTYPE**: True if a device of any type is erroneous in the system. • **TC**: True if not all TCs have communication contact with all the other TCs in the system. • **SERVO**: True if any servo in the system is erroneous. • **PLC**: True if any PLC reporter device in the system is erroneous. • **FIELDBUS**: True if any fieldbus device in the system is erroneous. • **IO**: True if any I/O device in the system is erroneous. • **EXTCOM**: True if any external communication device in the system is erroneous. • **API**: True if any API device in the system is erroneous. • **IDDEV**: True if any ID device device in the system is erroneous. • **PRINTER**: True if any TC printer device in the system is erroneous. • **TOOLSNET**: True if any ToolsNet device in the system is erroneous.
STN_DEVTYPE_ERR	DEVTYPE_ERR_ TYPE	This variable indicates erroneous devices related to the station per device type category. It functions the same ways as the SYS_DEVTYPE_ERR variable with the differences that only the devices connected to a TC that belongs to this station are checked. **Note:** There is one exception to the above rule. The flag TC is set True also if this station has lost communication contact with the System TC (TC1).
DEVICE_ERR	DEVICENO_ERR_ ARR	This variable indicates erroneous devices in the system per device number. The variable is an array of 20 BOOL where each element represents the device with the number equal to the array index number. That is, if DEVICE_ERR[10] is True then the device with device number 10 is erroneous.

The following shared memory variable can be used for handling of process data:

Variable	Type	Use
FB_PROCDATA_STORED	BYTE	The PowerMACS system will write True to it when it has written anything to the Process Data output area of the fieldbus. The System will never write False to it. In order to detect when it is set you must clear it from inside the PLC.
CYCLDATA_VAR_UPDATED	BYTE	This input is set True when there are new cycle data available in CycleData_Var. It can be set False by the PLC program to be able to check when new cycle data has arrived.
UPDATE_EXTCO_2	BYTE	When this variable is written with a positive flank (0->1) all variables controlled by the EXTCO2_out area will be updated. The PLC program must reset it to 0 afterwards, not faster than one execution loop. See sub chapter Access to PLC data of Serial Communication.

A shared memory variable can be written to from the PowerMACS system as well as from the PowerMACS PLC.

FB_PROCDATA_STORED may be used in many different ways depending on type of application. For example: In a system where cycle data is generated after each cycle (also at emergency stop) and this data is automatically loaded to the fieldbus device one way of using it is as follows:

1. Have the PowerMACS PLC write FALSE to the variable FB_PROCDATA_STORED just before, or simultaneously with, the start signal.
2. Wait for the PowerMACS system to write TRUE to the variable FB_PROCDATA_STORED before indicating that the cycle is finished to the fieldbus master.

5.3.2 Bolt variables

The following inputs can be used to monitor the status of each individual bolt:

Variable	Type	Use
COMPERRORS	ARRAY[1..50] OF BYTE	Reports the most commonly reported bolt errors for each of the bolts. The respective bits have the following meaning (bit 0 is the least significant): • 0: (value 1) THM (Torque High Monitoring) • 1: (value 2) TLM (Torque Low Monitoring) • 2: (value 4) AHM (Angle High Monitoring) • 3: (value 8) ALM (Angle Low Monitoring) • 4: (value 16) OK (Bolt status OK). This bit is set if the bolts final status is OK. • 5: (value 32) Reserved • 6: (value 64) Reserved • 7: (value 128) Reserved

Use the below output signals to control the behavior of the bolts:

Variable	Type	Use
BOLTCONTROL	ARRAY[1..50] OF BYTE	This output controls which bolts to run when the station is started. Takes one of the following values: • 0: Connect normally. The bolt will run the tightening program specified in the mode table when station is started. • 1: Disconnected with OK status • 2: Disconnected with NOK status • 3: Do not run during next cycle • 100 + Pgm. No: The bolt will run the tightening program with program number Pgm. No when station is started and mode table value is “Program from PLC” for the bolt. PgmNo is a number between 1 and 150. The difference between "Disconnected" and "Do not Run" is that in the first case the bolt will be reported with the given status and in the latter nothing is reported for the bolt. In neither case will the bolt start when the station is started.

The outputs BOLTCONTROL and DATAHOLD are useful when running a stitching application, i.e. when a single spindle is used to tighten more than one bolt. To generate one cycle data result, including only one station header, even though several starts are done follow the below scheme:

1. Set DATAHOLD = True
2. Set BOLTCONTROL = 0 or BOLTCONTROL = 100 + Pgm. No for all bolts to run in this cycle and BOLTCONTROL = 3 for all bolts that should not.
3. Generate a START. This will start only the selected bolts.
4. Wait for the cycle to end (monitor for example STATIONSTATUS)
5. If not all bolts are tightened yet go to step 2
6. Set DATAHOLD = False to release the cycle data, complete with one record for the station and one for each bolt tightened

The following shared memory variable also reflects the state and status of the bolts. Even though they are defined as shared memory variables, which mean that the PLC application can write to them, they should be used as inputs only.

Variable	Type	Use
BOLTGROUPS	BOLT_GROUPS_ARR	This is an array of 50 WORDS (16 bit unsigned integers) where each element represents the groups that a particular bolt belongs to. The respective bits in BOLTGROUPS[x] have the following meaning (bit 0 is the least significant): • 0: (value 1) Bolt x belongs to group 1 • 1: (value 2) Bolt x belongs to group 2 • 2: (value 4) Bolt x belongs to group 3 • 3: (value 8) Bolt x belongs to group 4 : • 8: (value 256) Bolt x belongs to group 9 • 9: (value 512) Bolt x belongs to group 10 • 10 -15: Reserved
BOLTSTATUS_EXT	BOLT_STATUS_EXT_ ARR	This variable is an array of 50 elements of type BOLT_STATUS_EXT_TYPE. Each element indicates the status of the last tightening for a bolt. The BOLT_STATUS_EXT_TYPE is a structure containing the following BOOL flags: • **OK** True if result is OK or OK after RM (OKR). • **OKRM** True if result is OK after RM. • **NOK** True if result is Not OK (NOK), Not OK due to RM (NOKRM) or Stop and Terminate Not OK (TERMNOK). • **NOKRM** True if result is Not OK due to RM (NOKRM). • **TERMNOK** True if result is Stop and Terminate Not OK (TERMNOK). The value of this variable is only changed as a result of running a cycle. It is updated as follows: • When the station is started all bolts that are set to be Disconnected OK/NOK, either using the PLC output BOLTCONTROL or the ToolsTalk PowerMACS forms "Mode Table" or "Bolt Set Up", the corresponding OK or NOK flag is set immediately. For all other bolts all status flags are cleared. • When the cycle is ready the relevant flags are set. Use this variable together with BOLTSTATE as an alternative to the earlier BOLTSTATUS input.

Variable	Type	Use
BOLTSTATE	BOLT_STATE_ARR	This variable is an array of 50 elements of type BOLT_STATE_TYPE. Each element indicates the current state of a bolt. The BOLT_STATE_TYPE is a structure containing the following BOOL flags: • **DONOTRUN** True if the bolt is ordered not to run, either using the PLC output BOLTCONTROL or the ToolsTalk PowerMACS "Mode Table" form. • **DISCON_OK** True if the bolt is disconnected with OK status, either using the PLC output BOLTCONTROL or the ToolsTalk PowerMACS forms "Mode Table" or "Bolt Set Up". • **DISCON_NOK** True if the bolt is disconnected with NOK status, either using the PLC output BOLTCONTROL or the ToolsTalk PowerMACS forms "Mode Table" or "Bolt Set Up". • **IDLE** True if the bolt is connected normally and is idle, that is, it is not disconnected or ordered not to run and is not executing a cycle. • **RUNNING** True if the bolt is running a cycle, that is, it is not idle. This includes running reject management, evaluating and waiting for PLC as well. • **RUNNING_RM** True if the bolt is running reject management. • **EVALUATING** True if the bolt is evaluating its monitoring results. • **WAITING_FOR_PLC** True if the bolt is waiting for a new START signal due to that it is executing a step that requires a this to continue. **Note:** Since the connect state (Do not run, Disconnected OK, Disconnected NOK) of a bolt may depend on the MODE (the Mode Table form), changing the value of the MODE input may affect this variable. This means that BOLTSTATE[x] always will reflect whether or not bolt x is disconnected or not, regardless from where it is disconnected. Use this variable together with BOLTSTATUS_EXT as an alternative to the earlier BOLTSTATUS input.

5.3.3 ID device variables

The following input and outputs can be used to control and monitor the Stations Work piece identifier function. See chapter: Advanced Station Settings for this is set up.

Inputs:

Variable	Type	Use
IDSTATUS	INT	Status for ID device read or write operation: • 0: Idle and OK • 1: Idle and NOK • 2: Busy Note that the read ID string is not valid until this signal is 0. After issuing an IDREAD you should always wait one PLC scan before evaluating IDSTATUS.
CURRIDSTRING	STRING(40)	This input always reflects the ID string known by the station. Will be included in the cycle data as the station level variable "Wp ID".
IDCODE	INT	Code from conversion of read ID string. See chapters: Conversion table for how the conversion is defined.

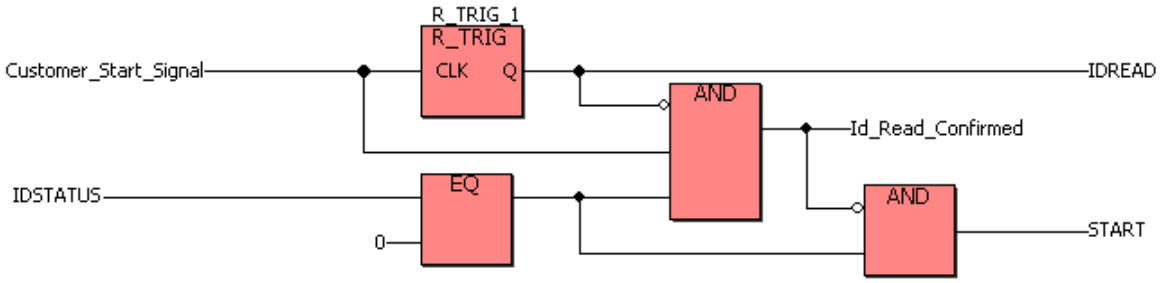
Outputs:

Variable	Type	Use
IDREAD	BOOL	Set to True to order reading of ID device connected to the station.
IDWRITE	BOOL	Set to True to replace the ID string currently known by the station with the value of IDSTRING. If the station is connected to an ID device this device will be used to generate a corresponding IDCODE. The written value will be echoed back to CURRIDSTRING.
IDSTRING	STRING(40)	String to replace the ID string currently known by the station. Sampled on the positive edge of IDWRITE.

The station will only use an ID string once. When it starts a cycle it will clear its internal ID string, and therefore also CURRIDSTRING, and then set the value of IDCODE to –32768.

Depending on which type of ID device that is used a read operation may take differently long time. For a barcode reader the last string scanned by the operator is stored internally by PowerMACS system so completing a read is done within less a few milliseconds. However, for devices of Escort memory, where the external data tag is read using a communication protocol, a read operation may take much longer time. You should therefore always check the input IDSTATUS to make sure that a read string is available to the station before you issue a start order.

Example: The below code issues an IDREAD order when the signal "Customer_Start_Signal" is set True, waits for the string to be returned, and then starts the tightening cycle.



5.3.4 Multiple identifier variables

The following input and outputs can be used to control and monitor the Stations Multiple identifiers function. See chapter: Advanced Station Settings for a description on how it is configured and works.

Inputs:

Variable	Type	Use
IDTYPE_1_STR ... IDTYPE_4_STR	STRING(40)	The value of the last accepted string for the respective Identifier Type. Cleared when on a positive edge of START.
IDTYPE_1_CODE	INT	Code generated by the Conversion table defined for Identifier Type 1.
IDTYPE_WO_STATE	BYTE	Work Order State. A number between 0 and 4 indicating the latest accepted item in the Work Order list. One (1) means that the first Work Order condition is met, two (2) that the second one was accepted, and so on. Set zero (0) when the Work Order sequence is reset.
IDTYPE_WO_COMPL	BOOL	Work Order Complete. Set True when all Types of the Work Order are accepted. Set False when a cycle is started or if the Work Order sequence is restarted.
IDTYPE_1_STS... IDTYPE_4_STS	INT	Status of the read operation of the ID device used as source for the respective Identifier Type: • 0: Idle and OK • 1: Idle and NOK • 2: Busy – waiting for the device to respond After issuing an IDTYPE_x_RD you should wait one PLC scan before evaluating IDTYPE_x_STS.

Outputs:

Variable	Type	Use
IDTYPE_1_RD ... IDTYPE_4_RD	BOOL	Set to True to order a read of the ID device configured as Source for the respective Identifier Type. Note! Issuing a read order using one of these outputs does not automatically lead to that the returned string is assigned the corresponding Identifier Type. This matching is controlled solely by the Multiple identifiers function.
IDTYPE_LOCK	BOOL	While True all input to the Multiple identifiers function is locked. All data received while lock is active is dropped.
IDTYPE_RESET_ALL	BOOL	Resets the Multiple identifiers function on a positive edge. This will clear all Identifier Type strings (IDTYPE_x_STR) and set the Work Order sequence to its initial state (IDTYPE_WO_STATE = 0).
IDTYPE_RESET_LAST	BOOL	A positive edge resets the last accepted Identifier Type. This will clear the string corresponding to the Identifier Type (IDTYPE_x_STR). If the Identifier Type is a part of a Work Order then the state of the Work Order is backed one step. If the last accepted Type was made the Work Order completed it will become *active* again, waiting for the same Type.
IDTYPE_WO_BYPASS	BOOL	A positive edge will accept the Type currently monitored by the Work Order. The Type will be assigned an empty string and a warning event is generated to indicate that a Work Order Type was bypassed.
PLC1_IDTYPE_STR ... PLC4_IDTYPE_STR	STRING(40)	String that can be used as Identifier Type Source. Sampled on the positive edge of PLCx_IDTYPE_WR.
PLC1_IDTYPE_WR ... PLC4_IDTYPE_WR	BOOL	A positive edge samples a new value of the corresponding PLCx_IDTYPE_STR variable.

5.3.5 GM DeviceNet variables

The following shared memory variables serve as the discrete interface to the GM DeviceNet device. For a more detailed description of the GM DeviceNet device see chapter: GM DeviceNet.

Even though the following variables are defined as shared memory type, which mean that the PLC application can read and write to them, they should be used as outputs only (seen from the PLC):

Variable	Type	Use
GM_IN	GM_INPUT_TYPE	This variable represents the discrete inputs to the GM DeviceNet device. GM_INPUT_TYPE is a structure containing the following BOOL flags: • **PARSET1** Parameter set code (1-8) mapped to Mode bit 0 (value 1). • **PARSET2** Parameter set code (1-8) mapped to Mode bit 1 (value 2). • **PARSET3** Parameter set code (1-8) mapped to Mode bit 2 (value 4). • **REVERSE** Reverse tool cycle. Uses mode 9 for the reverse program. • **START** A transition from FALSE to TRUE starts a tightening cycle. Positive edges while a cycle is running or when STOPDISABLE is True are ignored.. • **EPRIN** TRUE indicates that a new cycle may be started. PARSETx are sampled on a positive edge of this signal. • **STOPDISABLE** TRUE stops the current cycle and inactivate the spindles from any future cycle. Must be low to permit a new cycle to start. • **DISABLEDNETINP** When TRUE all inputs via DeviceNet except for the PVI number are disabled. **Note!** This input is always read, regardless of the value of the device parameter **Through discrete I/O** (see chapter: GM DeviceNet).

Even though the following variables are defined as shared memory type they should be used as inputs only (seen from the PLC):

Variable	Type	Use
GM_OUT	GM_OUTPUT_TYPE	This variable represent the discrete outputs from the GM DeviceNet device. GM_OUTPUT_TYPE is a structure containing the following BOOL flags: • **GREENLIGHT** Pass through signal from the DeviceNet input interface. (Green stack light). • **YELLOWLIGHT** Pass through signal from the DeviceNet input interface. (Yellow stack light). • **REDLIGHT** Pass through signal from the DeviceNet input interface. (Red stack light). • **ALARMHORN** Pass through signal from the DeviceNet input interface. (Alarm Horn). • **PARSET1** Parameter set code (1-8) confirmation. Mapped to Mode bit 0 (value 1). • **PARSET2** Parameter set code (1-8) confirmation. Mapped to Mode bit 1 (value 2). • **PARSET4** Parameter set code (1-8) confirmation. Mapped to Mode bit 2 (value 4). • **GLOBALACCEPT** Global accept (Set to 1 if the cycle completed OK). • **INCYCLE** Spindle in cycle, set to 1 when a cycle is running and to 0 when idle. • **FAULTRELAY** System fault relay (currently not used). • **CYCLECOMP** Rundown complete. Is TRUE when a cycle is ended and FALSE when running. • **EPROUT** Reflects the current value of the input Error Proofing Ready (EPR). • **LIGHT**(16) This is an array of 16 x 3 BOOL that indicates the status of the tightening for each of the bolts in the station. The status of each bolt is indicated using the following three BOOL signals: **GREEN** Good Rundown (Green light) **RED** Remove, Inspect & Repair Fastener (Red light) **YELLOW** Low Torque (Yellow light)

5.3.6 DC PLUS variables

The following variables are used for control of a DC PLUS Ethernet device. For a more detailed description of the DC PLUS Ethernet device see chapter: DC PLUS.

Inputs (seen from the PLC):

Variable	Type	Use
PLUS_CONNECTED	BOOL	TRUE if a connection to the PLUS server is established.
PLUS_DATAAVALIABLE	BOOL	TRUE if there are data available in the cycle data queue for the PLUS device.
PLUS_MODE	INT	Mode number from conversion of TMU. The conversion table in the PLUS device will be used for the conversion. The string to convert is a TMU received from PLUS, if “Station Type” on the PLUS device is SAW/WIS. Otherwise the string PLUS_OVERRIDETMU will be used. The value is updated when PLUS_GETTMU_OK is set to TRUE.
PLUS_GETTMU_CHECKED	BOOL	Reflects the selection of Station Type in the PLUS device in ToolsTalk PowerMACS. Set to TRUE for station type SAW and to FALSE for the other types. Only available for backward compatibility reasons, it is recommended to use PLUS_STATIONTYPE instead.
PLUS_STATIONTYPE	INT	Reflects the selection of Station Type in the PLUS device in ToolsTalk PowerMACS. Possible values are: 1 = SAW/WIS 2 = SPS 3 = VBA 4 = SSE
PLUS_SENDDATA_NOK	BOOL	Is set to NOK if a send of cycle data fails for some reason, e.g. bad VIN number, bad QO, etc. If there are more data in the cycle data queue they will be sent even if one send failed. PLUS_SENDDATA_NOK will be set until a positive edge is sent to PLUS_RESETSENDDATA_NOK.
PLUS_BUSY	BOOL	The device is busy sending a request for TMU or a Test telegram. Only tests started from the PowerMACS PLC, with PLUS_CHECKCOM will affect PLUS_BUSY.
PLUS_GETTMU_OK	BOOL	TRUE if the latest request for TMU succeeded.
PLUS_GETTMU_NOK	BOOL	TRUE if the latest request for TMU failed.
PLUS_CHECKCOM_OK	BOOL	TRUE if the latest test of the communication, started from the PowerMACS PLC with PLUS_CHECKCOM, succeeded.

Variable	Type	Use
PLUS_CHECKCOM_NOK	BOOL	TRUE if the latest test of the communication, started from the PowerMACS PLC with PLUS_CHECKCOM, failed.

Outputs seen from the PLC:

Variable	Type	Use
PLUS_CHECKCOM	BOOL	On a positive edge a test telegram “TSTS” is sent to the PLUS server. PLUS_BUSY will be set to TRUE and PLUS_CHECKCOM_OK and NOK will be set to FALSE. Then the test has finished PLUS_BUSY will be FALSE and PLUS_CHECKCOM_OK or NOK will be TRUE, depending on the result of the test. Has the same functionality as the Test button in ToolsTalk PowerMACS.
PLUS_GETTMU	BOOL	On a positive edge the value in “PLUS_BARCODE” is sent to PLUS in a request for a TMU and a PN or KN. PLUS_BUSY will be TRUE during the transfer and PLUS_GETTMU_OK will be TRUE when the result is ready for usage. The result of the request will be placed in CURRIDSTRING (the PN or KN) and PLUS_MODE (mode number from conversion of received TMU). CURRIDSTRING will have the letters ‘PN’ or ‘KN’ added as a prefix to the string received from PLUS. If “Station Type” in the PLUS device is different from SAW/WIS nothing is sent to PLUS. Instead the string in PLUS_BARCODE is copied directly to CURRIDSTRING with the letters ‘PN’ or ‘KN’ added as prefix. The string in PLUS_OVERRIDETMU will be sent to the conversion table to get PLUS_MODE. In a VBA station nothing at all will happen when a positive edge is sent to PLUS_GETTMU.
PLUS_BARCODE	STRING (10)	Id string that is sent to PLUS to get a TMU and PN or KN. If the string is treated as a production number “PN”, a body number “KN” or a carrier Id “LI” is controlled by the parameter “Barcode scanned” in the PLUS device in ToolsTalk PowerMACS. PLUS_BARCODE is sent on a positive edge on PLUS_GETTMU and the result can be used when PLUS_GETTMU_OK is set to TRUE. The returned PN or KN will be copied to CURRIDSTRING with the letters PN or KN added in the beginning. This string will be included in the cycle data as "Wp ID". The TMU received from PLUS will be converted to a mode number using the conversion table in the PLUS device. The result will be placed in PLUS_MODE.

Variable	Type	Use
PLUS_DOKUENABLE	BOOL	Set high to send Cycle data to PLUS. If the signal is high when a new Cycle data is produced it will be sent immediately to PLUS. If the signal is low when a new Cycle data is produced the new data will be stored in the Cycle data queue for the PLUS device. When the signal goes high the whole cycle data queue will be sent to PLUS. PLUS_DOKUENABLE is only used if “Send cycle data automatically” is not checked in the PLUS device in ToolsTalk PowerMACS. If “Send cycle data automatically” is checked all new cycle data will be sent as soon as the cycle has ended.
PLUS_QO	INT	The value of PLUS_QO will be used as station identification QO when sending requests for TMU to PLUS. PLUS_QO will only be used if “Use QO from PLC” is checked in the PLUS device; otherwise the value written in “QO for request of TMU” in the PLUS device will be used instead. When sending cycle data to PLUS the value in “Station QO” in the result will be used as QO.
PLUS_DELETE	BOOL	On a positive edge the latest cycle data is removed from the Cycle data queue for the PLUS device used for the station.
PLUS_DROP	BOOL	If PLUS_DROP is high when cycle data is stored for a cycle no data is written in the Cycle data queue for the PLUS device. The PLC variable CYCLEDATASTORED can be used to know when the data has been stored.
PLUS_FLUSH	BOOL	On a positive edge the whole cycle data queue for the PLUS device is deleted.
PLUS_RESETSENDDATA_NOK	BOOL	On a positive edge the variable PLUS_SENDDATA_NOK is set to FALSE.
PLUS_SEND_SONL	BOOL	On a positive edge the telegram SONL is sent to PLUS.
PLUS_SEND_SOFF	BOOL	On a positive edge the telegram SOFF is sent to PLUS.
PLUS_MANUALMODE	BOOL	Set to TRUE to tell the TC that the station is operating in manual mode. In manual mode the TC sends NNOT as answer to every telegram received from PLUS. No cycle data is sent to PLUS when PLUS_MANUALMODE is high. Only used if the station type is VBA.
PLUS_OVERRIDETMU	STRING (25)	String that will be sent to the conversion table in the PLUS device to get a mode number. The resulting mode number is found in PLUS_MODE. The string in PLUS_OVERRIDETMU will only be used if “PowerMACS requests TMU” on the PLUS device is unchecked. Otherwise the TMU received from PLUS will be used.

5.3.7 Globals

These variables are used to transfer information from one station PLC to another station PLC. All stations can reach these global variables.

Variable	Type	Use
SYSTEMGLOBAL	INT	For communication between PLCs on different stations.

The variables are declared as a shared memory and can therefore both be read and written to both by the PowerMACS System and the PowerMACS PLC.

You should see to that only one station write to a given variable but any number of stations can read it.

5.3.8 Bolt specific commands

The bolt specific commands enables extended control of individual bolts using the following input and outputs variables.

Outputs:

Variable	Type	Use
BOLTCTRL2_CMD	INT	The command to execute on the bolts selected using BOLTCTRL2_BOLTSEL. See table below for supported command codes. The command is sent to the station when BOLTCTRL2_CMD changes to a non zero value.
BOLTCTRL2_BOLTSEL	ARRAY[1..50] OF BYTE	Assign a non zero value for the bolts that should execute the BOLTCTRL2_CMD command. **Note 1** The array index corresponds to the bolts ordinal number within the station (not the user defined). **Note 2** The highest element (bolt) checked is NO_OF_BOLTS.

Currently the following command codes are supported:

Mnemonic	Value	Description
DROPLASTCD	1	Drop the last cycle data stored for the bolts selected. The command may only be issued when the station is idle, that is, not executing a cycle. The current cycle data must also include at least one result record for each of the bolts selected. In addition to the general result values the following may also be returned in BOLTCTRL2_BOLTRES[x]: • 5: NODATATODROP. There is no data to drop for the bolt. • 6: CDCORRUPT. Cycle data is found to be corrupt. Unless the command was executed OK for all bolts the cycle data is left untouched. **Note!** The value of the station level result variable Status is always set according to the status of all individual bolt results in the cycle data. **If used erroneously this function might cause NOK parts to be reported as OK parts! Removing all NOK results from a cycle data produced by a station causes it to be reported as OK. You should therefore not use this function unless you are sure how to handle it.**

Illegal commands, errors and successful drops are logged using events.

Note! New command codes may be added in future versions. To avoid unexpected behavior when upgrading you should not make use of any other codes then the above listed and zero (0).

Inputs:

Variable	Type	Use
BOLTCTRL2_STS	INT	Overall execution status of the BOLTCTRL2 command: • 0: Idle and OK. Command completed OK for all bolts selected. • 1: Idle and NOK. Command failed for at least one of the bolts selected. Check BOLTCTRL2_BOLTRES for status of the individual bolts. • 2: Busy executing the command. **Note!** The operation is not complete until this signal is 0 (or 1 if it fails). After issuing a BOLTCTRL2 command you should wait one PLC scan before evaluating BOLTCTRL2_STS.
BOLTCTRL2_BOLTRES	ARRAY[1..50] OF BYTE	Returns the result per individual bolts of the command executed. The below values are common for all commands. See also the description of the commands for additional command specific return codes: • 0: OK. Command executed OK. • 1: ERROR. Command failed for unknown reason (that is, anyone not listed here or per command). • 2: NOBOLTSSEL. There where no bolts selected for the command to execute on. • 3: CMDNOTIMPL. Command not implemented (unknown value of BOLTCTRL2_CMD). • 4: STNNOTIDLE. Station is not idle (many commands may only be executed when the station is not running a cycle). **Note!** Values are only returned for the bolts selected using BOLTCTRL2_BOLTSEL.

Using the BOLTCTRL2 variables in combination with the output DATAHOLD it is possible run reject management from the PowerMACS PLC, or an external PLC, and still report one complete cycle data for the tightening using PowerMACS standard reporting facilities.

Example: This example shows how reject management can be handled by the PowerMACS PLC for a station having four bolts named "B1", "B2", "B3" and "B4". The scenario is that all four bolts starts by running program "Tighten" (mode 1), bolt "B1" and "B2" fails and are loosened by program "Loosen" (mode 2) and then retightened OK by running program "Tighten".

1. Set PLC output DATAHOLD to TRUE.
2. Set MODE to 1 and start a cycle. Wait for it to be completed. Say that bolt "B1" and "B2" finished NOK.
3. To drop the NOK result of the failing bolts "B1" and "B2" do the following: First set up BOLTCTRL2_BOLTSEL as below:

   BOLTCTRL2_BOLTSEL[1] = 1
   BOLTCTRL2_BOLTSEL[2] = 1

BOLTCTRL2_BOLTSEL[3] = 0
BOLTCTRL2_BOLTSEL[4] = 0

Then order the station to remove the last result stored for the selected bolts by changing BOLTCTRL2_CMD from 0 to 1.

Wait for BOLTCTRL2_STS to become 0 (idle OK). This indicates that the station has completed the drop command.

4. To loose and retighten only the failing bolts the OK bolts must be inhibited. Therefore set up BOLTCONTROL as below:

   BOLTCONTROL[1] = 0 (* Connected normally *)
   BOLTCONTROL[2] = 0 (* Connected normally *)
   BOLTCONTROL[3] = 3 (* Do not run *)
   BOLTCONTROL[4] = 3 (* Do not run *)

5. Loose the bolts by setting MODE to 2 and start the cycle. Wait for the cycle to be completed.
6. Remove the result of the loosening cycle by repeating step 3 above.
7. Retighten bolt "B1" and "B2" again using MODE 1 and starting the cycle. Wait for the cycle to be completed.
8. Given that the complete tightening now is considered done, either since the retightening of bolt "B1" and "B2" ended OK, or that no more attempts are allowed, releases the produced cycle data by setting DATAHOLD to FALSE.

The cycle data produced using the above sequence will the be:

```
Time: 2005-07-08 13:00:00
Status: OK
Bolt: B3   Program: Tighten   Status: OK
Bolt: B4   Program: Tighten   Status: OK
Bolt: B1   Program: Tighten   Status: OK   (the result from the retightening)
Bolt: B2   Program: Tighten   Status: OK   (the result from the retightening)
```

while the same operations without using the BOLTCTRL2 bolt data drop function would be:

```
Time: 2003-07-08 13:00:00
Status: NOK
Bolt: B1   Program: Tighten   Status: NOK  (the result from the first tightening)
Bolt: B2   Program: Tighten   Status: NOK  (the result from the first tightening)
Bolt: B3   Program: Tighten   Status: OK
Bolt: B4   Program: Tighten   Status: OK
Bolt: B1   Program: Loosen    Status: OK   (the result from the loosening)
Bolt: B2   Program: Loosen    Status: OK   (the result from the loosening)
Bolt: B1   Program: Tighten   Status: OK   (the result from the retightening)
Bolt: B2   Program: Tighten   Status: OK   (the result from the retightening)
```

5.3.9 Device commands

Using the device command input and outputs it is possible to control individual devices directly from the PLC.

Outputs:

Variable	Type	Use
DEVCMD_CMD	INT	The command to send to the device pointed out by DEVCMD_DEVNAME. See table below for supported commands. The command is sent to the device when DEVCMD_CMD changes to a non zero value.
DEVCMD_DEVNAME	STRING(20)	The name of the device, as defined in the ToolsTalk PowerMACS, to send the command to. For example, "Printer 01".
DEVCMD_CMDPARSIZE	INT	Size, in number of bytes, of additional command parameters specified using DEVCMD_CMDPARDATA[]. **Note!** Only specific commands make use of addition command parameters.
DEVCMD_CMDPARDATA	ARRAY[1..1000] OF BYTE	Additional command parameter data. Layout and usage depends on the targeted device and command. See table below for specification.

Currently only devices of type "Printer" are supported. For it the following command is available:

Mnemonic	Value	Description
PRINT	1	Prints DEVCMD_CMDPARSIZE bytes of the data supplied in the DEVCMD_CMDPARDATA area. The data is assumed to be binary and is printed without any additional formatting. That is, to print the string "ABC" the following decimal data should be set as parameters: DEVCMD_CMDPARDATA[1] := 65 DEVCMD_CMDPARDATA[2] := 66 DEVCMD_CMDPARDATA[3] := 67 and DEVCMD_CMDPARSIZE := 3.

Illegal device types as targets and/or command codes are logged using events.

Note! Support for other device types and command codes may be added in future versions. To avoid unexpected behavior when upgrading you should not make use of any other codes then the above listed and zero (0).

Inputs:

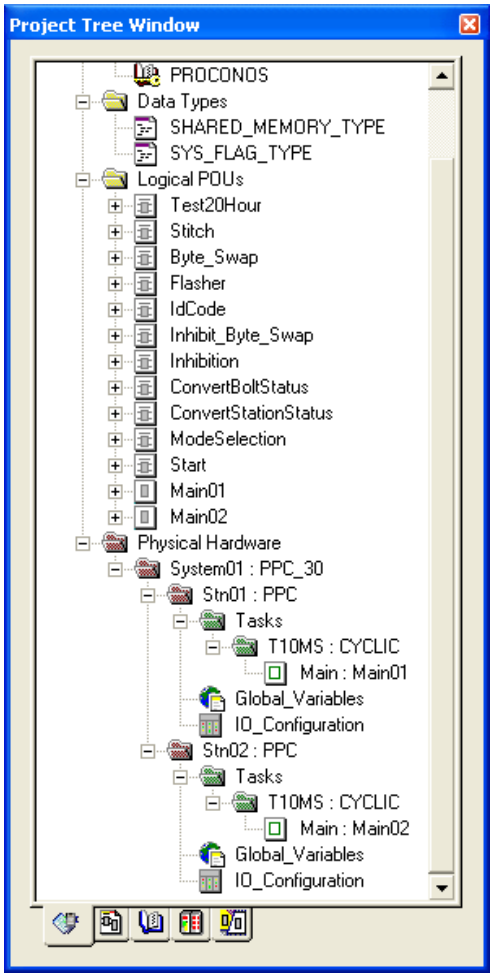
Variable	Type	Use
DEVCMD_STS	INT	Overall execution status of the DEVCMD command: • 0: Idle and OK. Command completed OK. • 1: Idle and NOK. Command failed. • 2: Busy executing the command. **Note!** The operation is not complete until this signal is 0 (or 1 if it fails). After issuing a DEVCMD command you should wait one PLC scan before evaluating DEVCMD_STS.

5.4 Programming the PLC

To edit the PLC program select the item **Program** from the **PLC** menu.

For every PowerMACS setup there is a PLC project. This project is created by the setup wizard, see **Creating a new system using the Set Up Wizard**, and is stored on disk together with the PowerMACS setup file. It has the same name as the setup file but the extension ".ZWT2" instead of "PM4".

For a two station system the PLC project tree displayed when the PowerMACS PLC editor is opened will look as follows:

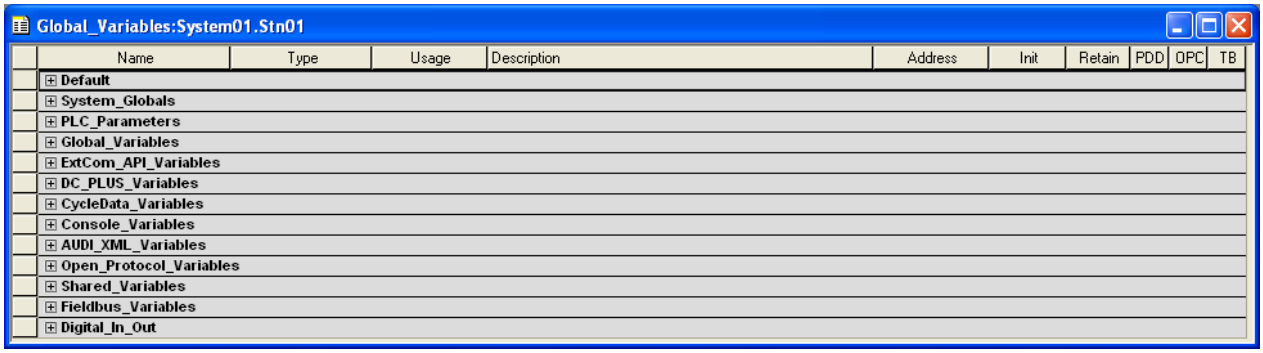


Each station in the system is represented in the PLC project as a Resource. A Resource corresponds to a physical unit that can execute a PLC program; in our case this is a station TC. They are named Stn01 for the first station in the system, Stn02 for the second, and so on.

To execute a program the Resource uses one or more tasks. A task controls the time scheduling of a program. Tasks are typically cyclic which means that they are executed periodically with a given time interval. The Setup Wizard creates one task named T10MS, which is executed every 100th millisecond, for each of the resources.

The program that is executed by a task is called POU, or Program Organization Unit. The Setup Wizard creates one POU for each station in the system. They are named Main01 for the first station, Main02 for the second, and so on. When created all Main programs looks the same since they are copied from the PowerMACS standard template. Normally you must adopt the program for each individual station. You do that by double clicking on the icon representing the variable worksheet (MainNNV) and/or the icon representing the logic worksheet (MainNN) of the program of interest.

Each resource has a number of variables that are global within the resource. These are defined on the worksheet named Global_Variables and can be used in all your programs (POUs) executed by the resource.

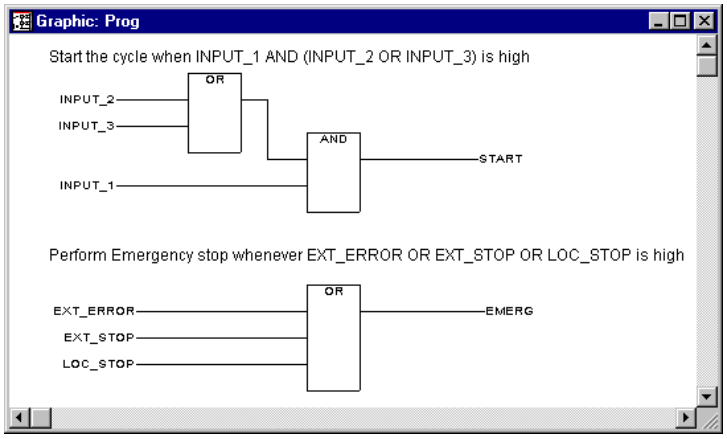


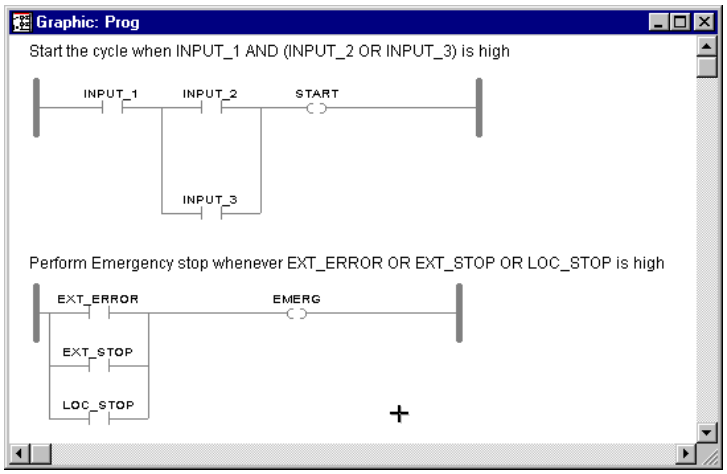
The variables defined on Global_Variables are divided in the following groups:

- **Default** and **Global_Variables**: - Consists of a number of PLC internal variables that can be used for advanced debugging of your PLC program.
- **System_Globals**: - This group contains all variables interfacing the PLC with the tightening controller part of the PowerMACS system. See **System Globals** for details.
- **ExtCom_API_Variables**: - This group contains variables that are mapped to optional external communication and/or API devices. See **Serial Communication** and **API, Application Programmers Interface** for how this is configured and used.
- **CycleData_Variables**: - This group contains variables that are mapped to the optional cycle data interface. See the description of **PLC** in chapter: Peripheral Devices, for how this is done.
- **DC_PLUS_Variables**: - This group contains all PLC variables used for interfacing a DC PLUS server. See the description of **DC PLUS** in chapter: Peripheral Devices, for how this is done.
- **AUDI_XML_Variables**: - This group contains all Audi XML device variables.
- **Shared_Variables**: - This group contains variables that are shared between the optional fieldbus interface and the digital I/O interface. See chapter: **Fieldbus Interface** for how this is done. See **I/O Device** for a detailed description.

- **Fieldbus_Variables**: - This group contains the fieldbus mapping. See chapter: **Fieldbus Interface**.
- **Digital_In_Out**: - This group contains variables only for digital I/O. Unlike the **Shared_Variables** group, these signals are not shared with fieldbus.

The programming of the PLC logic can be done in two ways: - using functional blocks or ladder logic. In both cases you use a graphical editor. Below is given two examples of this.





After editing the program you must compile it before it can be downloaded. This is done using the item Make in the menu Build. If your program contains any errors you will be informed about them.

When the program is successfully compiled it is possible to download to the target system. This is normally done as follows:

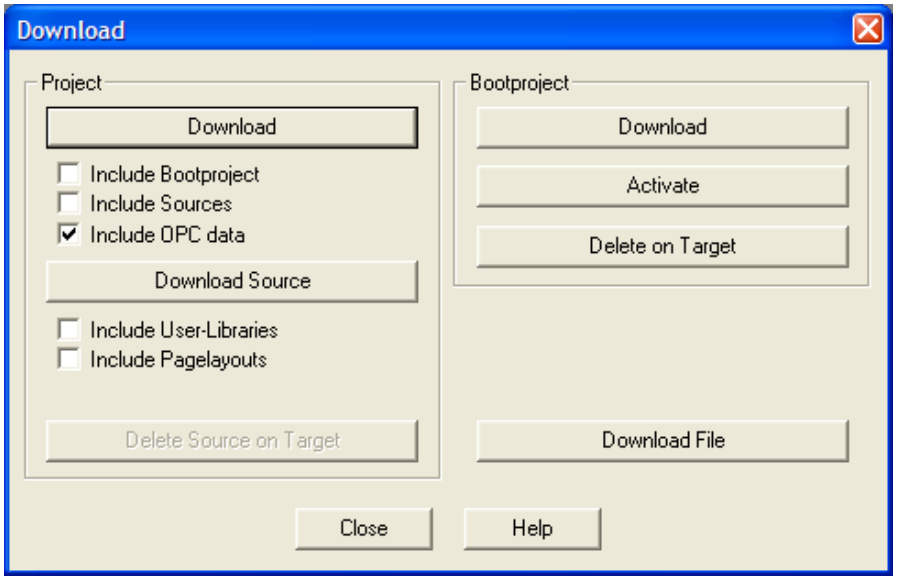
1. First close the PowerMACS PLC.
2. If the WinTC already is on-line download the PLC project by selecting the menu item **PLC – Synchronize**. During synchronization a status form is shown se Display PLC Status.
3. If the WinTC is not on-line then just go on-line

Downloading the PLC project this way ensures that the PLC program and the rest of the setup is kept synchronized.

However, during the development phase, when you have to change the program often, and quickly want to study the effect of your modifications you can download the program from inside the PLC. This is done using the following sequence:

- Open the **Project Control** dialog from the **Online** menu.

- Stop the execution of the PLC by pressing the **Stop** button.
- Reset the PLC by pressing the **Reset** button.
- Open the **Download** dialog by pressing the button **Download**.



- Press the **Download** button in the **Bootproject** frame
- Open the **Download** dialog again by pressing the **Download** button in the **Resource Control** dialog.

- Activate the Boot project by pressing the **Activate** button.
- Start the execution of the PLC by pressing the button **Cold** in the **Resource Control** dialog.

When your program is downloaded you can study current status of all signals in the program by using the **Debug** alternative in the PLC **Online** menu. If the program has not been compiled yet the PLC will complain by saying, "Cannot switch to online mode because the project is not successfully compiled!" In these cases just compile the program, and if there are errors, fix these. Then try to go on-line again.

Note 1: Saving the project from inside the PowerMACS PLC does **not** implicate cause the PLC project to be saved together with your setup on disk. To do that you must you choose **Save**, or **Save As**, from the WinTC **File** menu.

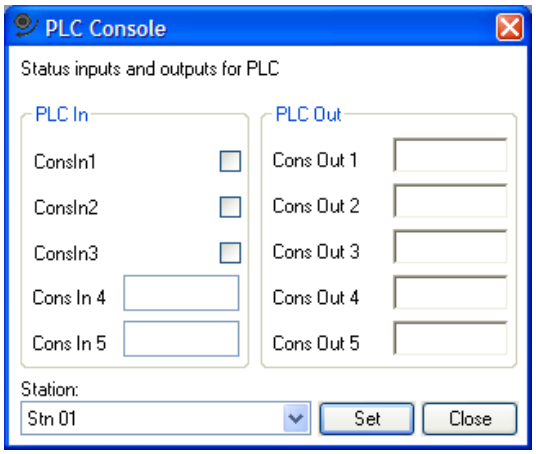
Note 2: It is the ZWT2 file that is regarded to be the source of the PLC project. It contains the project in compressed form. Whenever you open a PowerMACS setup the PLC project is expanded to the folder named **CurrentSetup**, which is located below the folder you installed PowerMACS in. Keep in mind that this folder only is used as a temporary working folder. Whenever you save the setup the PLC project is also compressed and saved to disk. When you go on-line and download a setup to a system the ZWT2 is also downloaded.

For more details on how to program the PLC see the MULTIPROG wt manual.

5.5 PLC Console

The *PLC Console* interface can be used as a simple operator's interface to your PLC application. It is invoked using the **PLC – Console…** item.

Use this form to display internal data of your PLC program that you want the operator to be able to see or modify.



Each of the controls in the frames PLC In and PLC Out are mapped to specific inputs and outputs in the PLC as described in the tables below. See also the Station variables chapter.

PLC In:	Type	Connected to PLC Input
Cons In 1	BOOL	CONS_IN_1
Cons In 2	BOOL	CONS_IN_2
Cons In 3	BOOL	CONS_IN_3
Cons In 4	STRING(40)	CONS_IN_4
Cons In 5	STRING(40)	CONS_IN_5

PLC Out:	Type	Connected to PLC output
Cons Out 1	STRING(40)	CONS_OUT_1
Cons Out 2	STRING(40)	CONS_OUT_2
Cons Out 3	STRING(40)	CONS_OUT_3
Cons Out 4	STRING(40)	CONS_OUT_4
Cons Out 5	STRING(40)	CONS_OUT_5

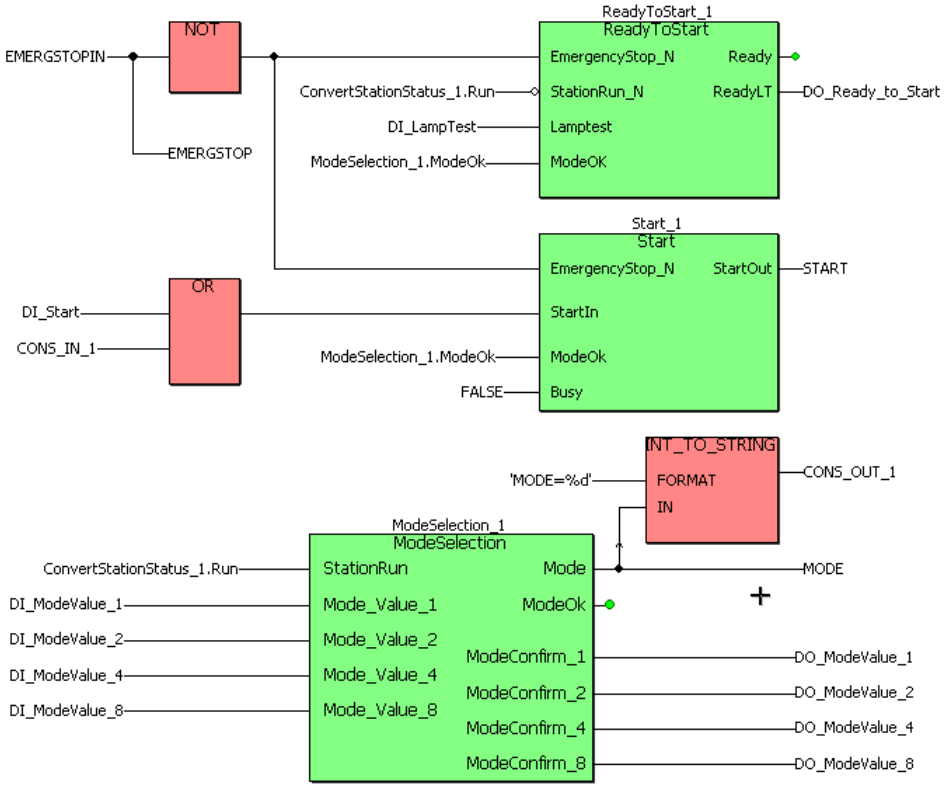
The outputs will be displayed automatically for the selected station (given by the **Station** combo box).

To set inputs do the following:

1. Enter the new values for one or more of the inputs. A checked check box corresponds to True, an unchecked to False.
2. Press the button Set. This cause the new values to be sent to the PLC.

Note! The set values will remain in the PLC until next time they are changed, or the PLC is restarted. Closing the PLC Console window will **not** clear any values.

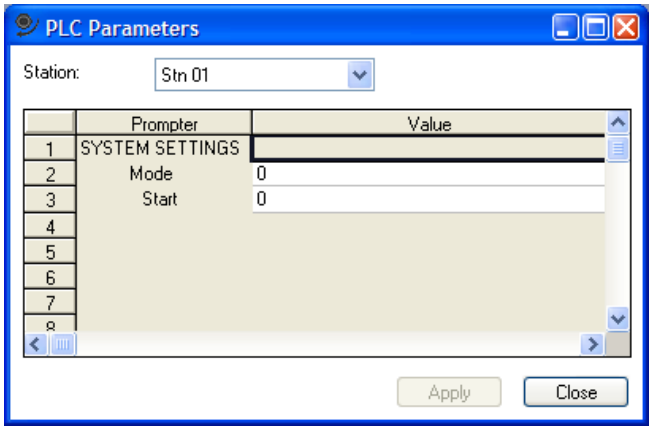
If you, as an example, connect "Cons In 1" to the PLC output START then you can simulate the external start signal that normally is input via some external device. Below the currently selected mode, which is set using digital inputs in this case, is also displayed as the "Cons Out 1" variable.



5.6 PLC Parameters

The PLC Parameters form, invoked using the **PLC – Console…** menu item, is similar to PLC Console interface in the sense that they both can be used as a simple operator's interface to your PLC application.

The difference is that the PLC Parameters are stored in non-volatile memory, and therefore will survive a restart of the PLC while all changes made using the PLC Console will vanish.



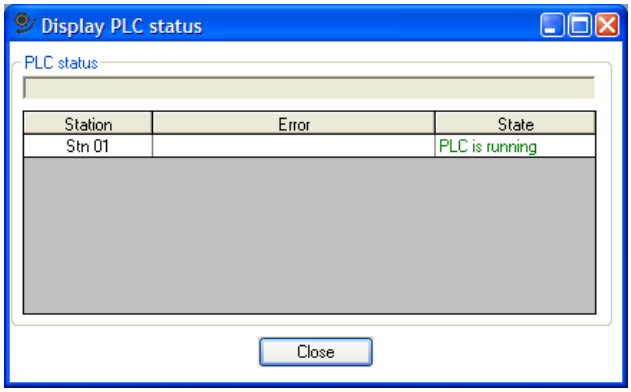
The parameters that are accessible using this form must first have been defined using the PLC Parameters Set Up form.

To change the value of a parameter just enter the new value in the Value column and press the **Apply** button.

5.7 Display PLC Status

The Display PLC status form, normally invoked using menu item **PLC – Display Status…**, is a tool for checking the status of the PLC runtime kernels when online.

This form is displayed automatically when the system is online and menu item **PLC – Synchronize …** is used.



The frame **PLC status** lists all PLC runtime kernels in the system. The list contains the name of the station and the state of the PLC runtime kernel.

Possible state for the PLC is:

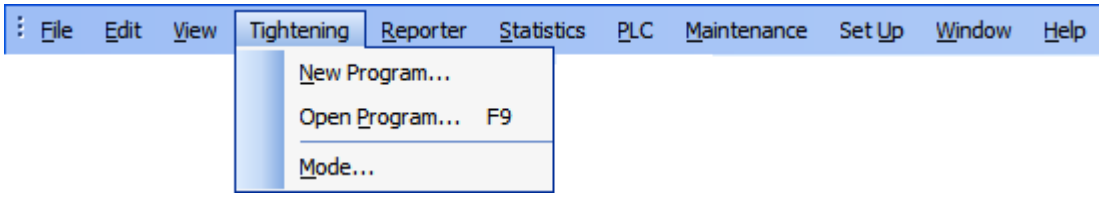
- Synchronizing new PLC program (Blue), synchronization has begun and the PLC is in an undefined state.
- PLC is running (Green), the PLC is in state running.
- PLC is stopped (Red), the PLC is stopped and waiting to start.

When the form is invoked by the **PLC – Synchronize** command it is only possible to close the form when the state of all PLC:s is PLC is running. During synchronization a progress bar is shown.

6 Tightening

6.1 Tightening - Overview

Most functions needed for to create and edit tightening programs are found in the **Tightening** menu



Select **New Program…** to Create a new Tightening Program.

To edit an existing program or sequence select **Open Program…** or **Open Sequence…**. Both open The Tightening Program form

Select **Mode…** to configure the mode table using The Mode Table form.

This chapter describes most things about the tightening process, for example

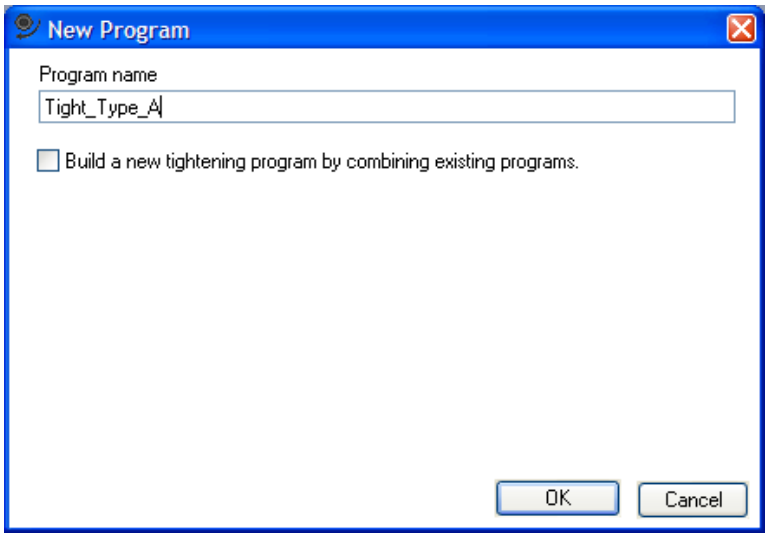
- How it works
- How to create a new program
- How to edit an existing program

To understand how the tightening process is actually performed within a PowerMACS system you should be familiar with all parts that make up the tightening process.

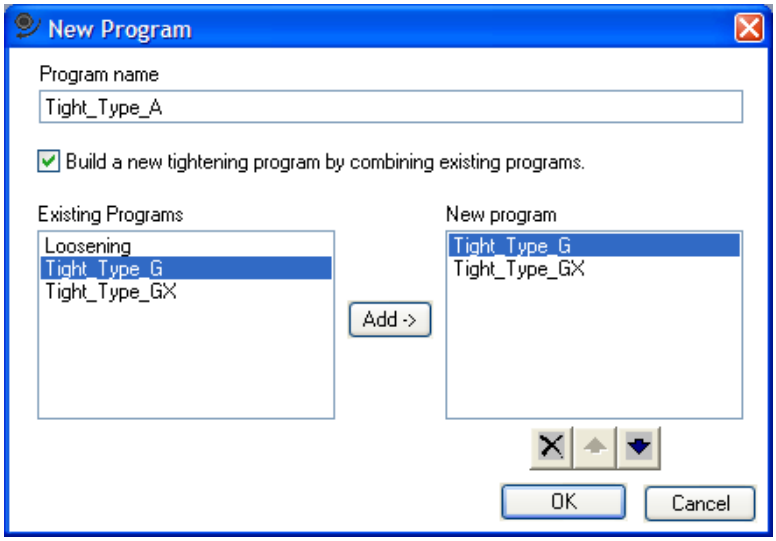
1. The PLC starts the tightening by setting outputs for start of tightening, for selection of which bolts to execute, and which tightening mode to run.
2. The Mode Table, set up using The Mode Table form, is used by the system to figure out which tightening program each bolt should run for the chosen mode.
3. The Tightening Program chapter describes how the tightening should be done for each bolt. A program is built using steps and sequences.
4. The spindles perform the actual tightening according to the information in the tightening program.

6.2 Create a new Tightening Program

The New Program wizard is opened using the **Tightening** - **New Program** menu or by right clicking on the **Programs** folder in the System Map and selecting **Add Program**.



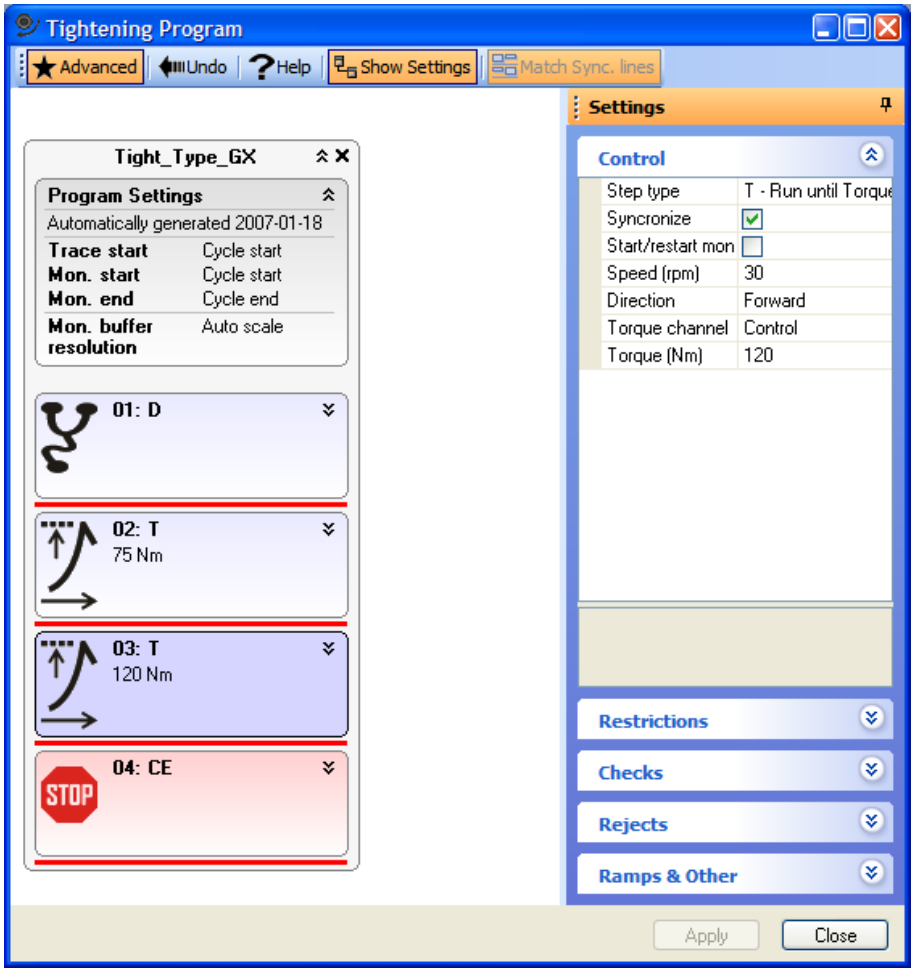
Start by entering a proper name for the new program in the **Program name** field. If you want to create a new program based on another program check the **Build a new tightening program by combining existing programs** checkbox.



When you are ready press **OK** to create the program. This add the program to the System Map tree.

6.3 The Tightening Program form

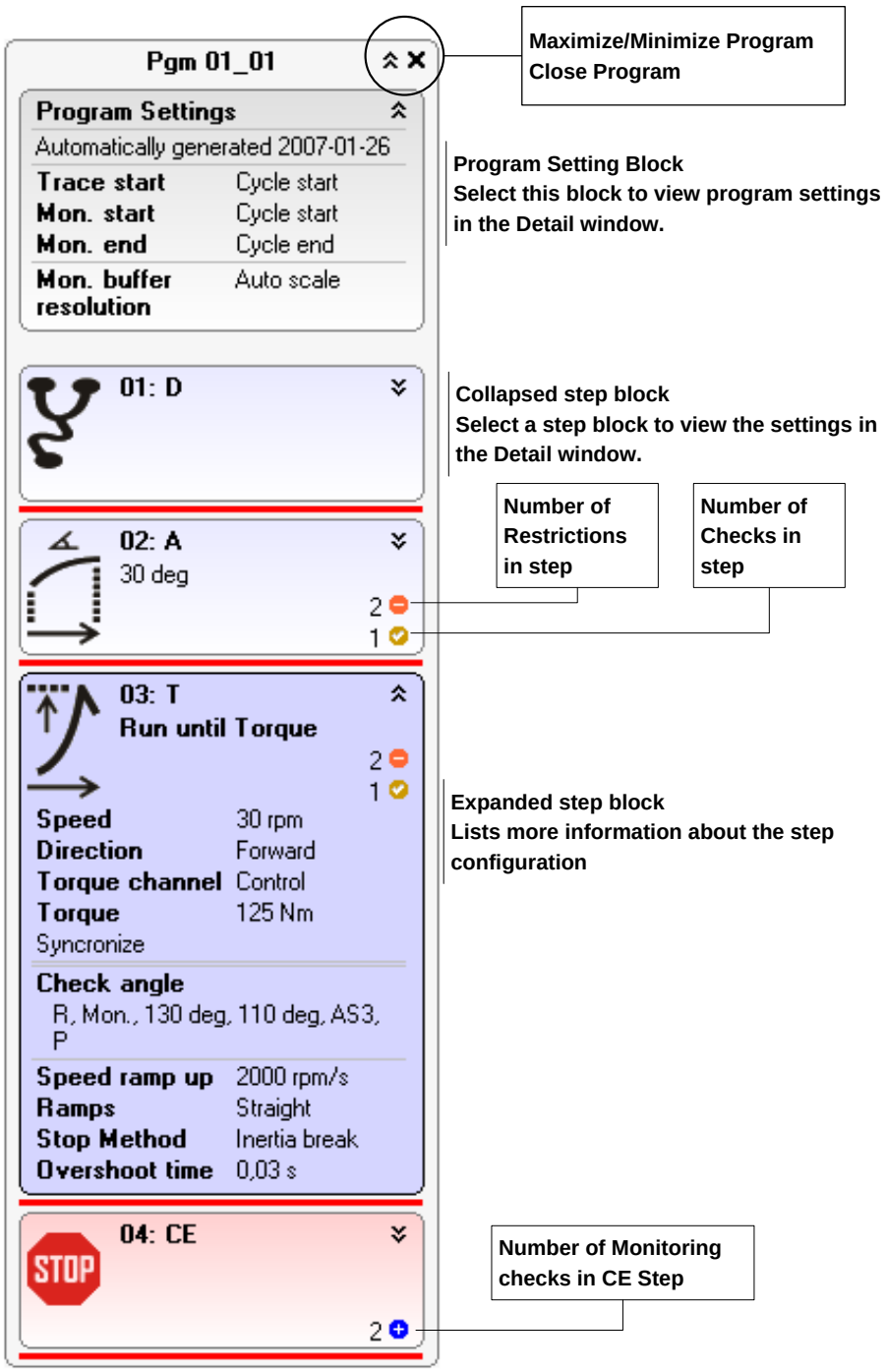
Tightening Program form is opened using the **Tightening - Open Program…** or by double clicking a program in the System Map. It is used for creation and editing on or several programs. A detailed description of programs is given in the Tightening Program chapter.



You can edit more than one program simultaneously by opening more programs using the System Map, it is also possible to drag & drop steps between programs.

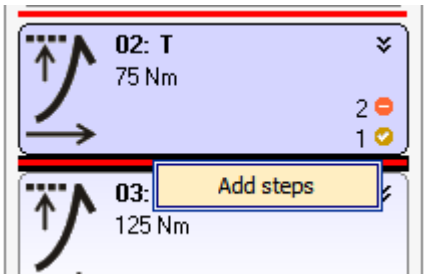
Programs are displayed on the left side of the Tightening Program form, to the right are the **Settings** for the currently selected item shown (this can be general program settings or details for a selected step).

Programs are made up of steps which you insert and remove by right clicking on a program or by dragging items from the **dynamic tool**.


Pgm 01_01
Maximize/Minimize Program
Close Program
Program Settings
Automatically generated 2007-01-26
Trace start Cycle start
Mon. start Cycle start
Mon. end Cycle end
Mon. buffer resolution Auto scale
Program Setting Block
Select this block to view program settings in the Detail window.
01: D
Collapsed step block
Select a step block to view the settings in the Detail window.
Number of Restrictions in step
Number of Checks in step
02: A
30 deg
2
1
03: T
Run until Torque
2
1
Speed 30 rpm
Direction Forward
Torque channel Control
Torque 125 Nm
Syncronize
Check angle
R, Mon., 130 deg, 110 deg, AS3, P
Speed ramp up 2000 rpm/s
Ramps Straight
Stop Method Inertia break
Overshoot time 0,03 s
Expanded step block
Lists more information about the step configuration
04: CE
STOP
2
Number of Monitoring checks in CE Step

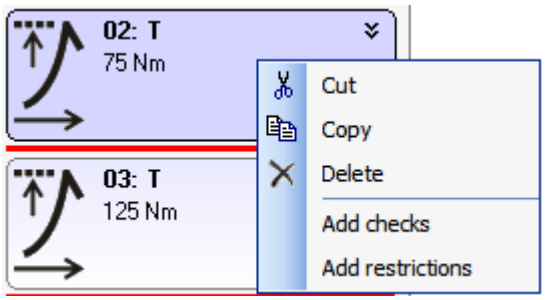
6.3.1 Tightening Program step operations

To add a step you can drag the step from the **Dynamic Tool** window and drop the step on the desired location in the program. It is also possible to right click between two steps and select **Add steps** from the popup menu.

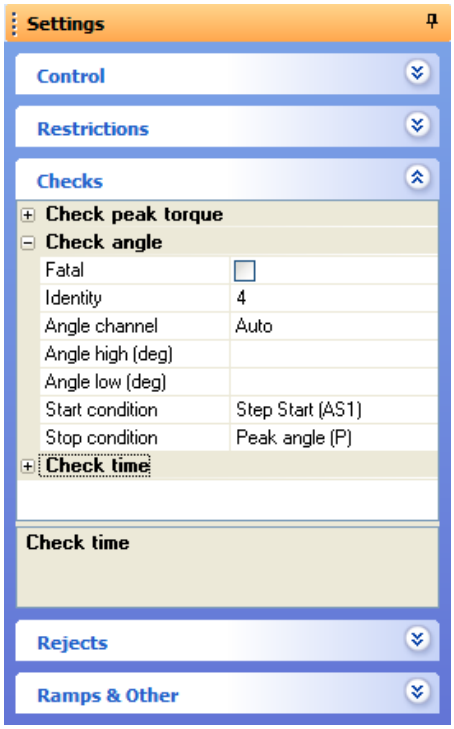


It is also possible to hold the **Ctrl** button while dragging an existing step from the same (or another) program to a new location – this will create an identical copy of the step. A copy operation is indicated by a small “plus sign” when the mouse is over the drop location.

When you right click on a step a popup menu is opened, here it is possible to **Cut**, **Copy** or **Delete** the step.



Select **Add checks** to add checks to the step and **Add restrictions** to add restrictions to the step. Note that it is also possible to drag & drop checks and restrictions to a step from the dynamic tool.



The **Settings** window displays parameters for the currently selected item (step or program). The default location is on the right side in the **Tightening Program** window but it can be placed freely.

The information for programs is divided in the following groups. To show program properties click on the topmost rectangle in the program.

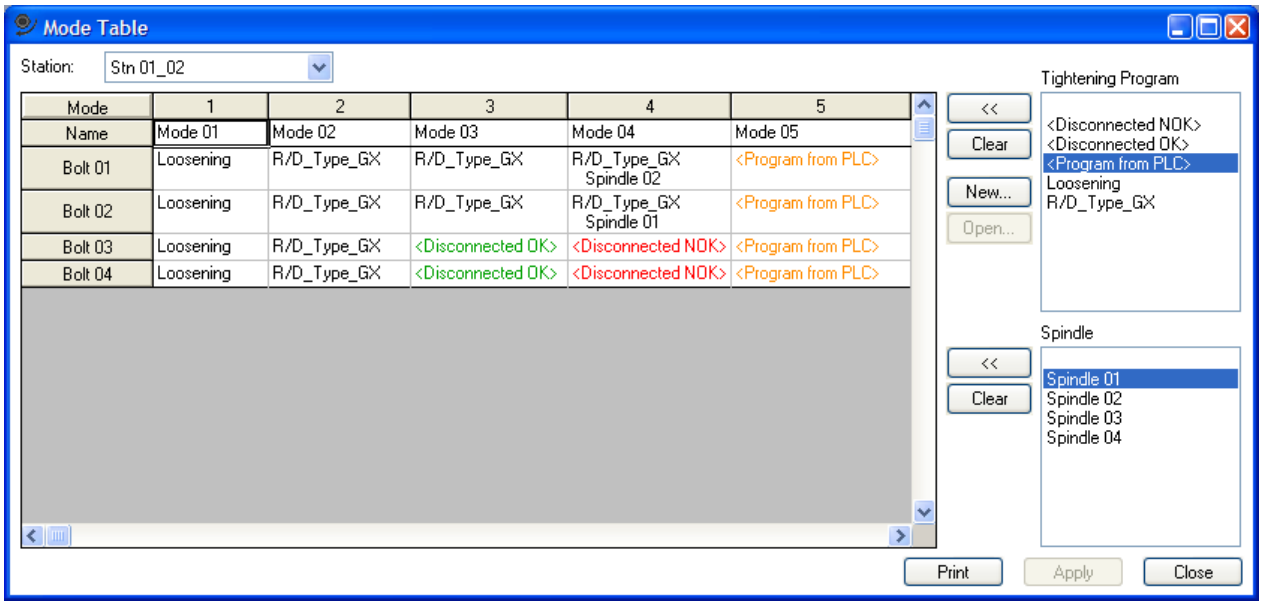
Group	Use
Common	Basic properties for the program
User Variables	Definition of program user variables and values.
Settings	Trace, monitoring and report limit settings for the program.

The information for steps is divided in the following groups. To show step properties click on a step in the program.

Group	Use
Control	What task the step performs. See Step – Control.
Restrictions	What restriction to run during the step. See Step – Restriction.
Checks	What checks to execute after the step. See Step – Check.
Rejects	Reject management in case anything goes wrong. See Step – Rejects.
Ramps & Other	Other configuration parameters for the step. See Step – Ramps & Other.

6.4 The Mode Table form

The Mode Table form is opened using the **Tightening - Mode…** menu item or by right clicking on the **Mode Table** for a station in the System Map and selecting **Open Mode Table**. The Mode Table form is used to set up which tightening program each bolt in a station should execute for a given mode.



Each station has its own Mode Table. Use the **Station** combo box to select the station for which the mode table should be edited.

The Mode Table has one column for each mode (up to 50) and a row for each bolt in the station. On the top row, labeled **Name**, you can specify a symbolic name for each mode. This name may be included in the cycle data as the station level result variable Mode.

The remaining rows are labeled with the names of the respective bolts. On these you should specify the tightening program to use for each mode. You may also specify which spindle to use to tighten the bolt in each mode. If no spindle is specified, the default spindle for the bolt is used (default spindle is defined in Bolt Setup).

The list labeled **Tightening Program** to the right lists all programs available in the system. In addition to these it also include the following entries which have special meaning:

- “Disconnected OK”
- “Disconnected NOK”
- “” (empty)
- “Program from PLC”

Bolts marked with Disconnected OK/NOK will be treated as disconnected for the corresponding mode. That is, they will not be executed but return a result, either OK or NOK depending on the selection. Disconnecting bolts here is equivalent to disconnecting them using the Bolt Status parameter of the Bolt with the difference that the selecting is only valid for specific modes.

Leaving a cell empty will have the effect that the bolt is not run when the station is started in the given mode. This corresponds to setting the PLC out BOLTCONTROL to the value 3, that is "Do not run", for the bolt. This may be very useful for so called stitching applications where the same spindle is used to tighten several bolts. See also the description of the PLC Bolt variables.

Bolts marked with Program from PLC will get the program to run from the PLC when the station is started in the given mode. The station will use the PLC out BOLTCONTROL to find the program for the bolt. It is essential that all programs used by Program from PLC have PgmNo between 1 and 150. Program from PLC may be very useful when the combinations of programs to run on different bolts are many and you want the PLC logic to determine what tightening program to run on different bolts. It is possible to combine program numbers with “Do not run” if stitching or reject management is to be handled. See also the description of the PLC Bolt variables.

You may type the program name directly in the cells, but it is probably easier to use the copy function to the right. It is used as follows:

1. Mark the program you want to copy in the **Tightening Program** list.
2. Mark the cells in the mode table you want the name copied to.
3. Press the << button next to the Tightening Program list.

To clear the selected Tightening Program from a number of cells mark them and press the **Clear** button next to the Tightening Program list.

If you want to Create a new Tightening Program, press **New…**.to open the New Program wizard.

To edit or view a program, mark it in the **Tightening program** list and press **Open…**. This will open The Tightening Program form.

The list labeled **Spindle** to the left lists all spindles available for the current station. The default spindle of all the bolts in a station belongs to the station. In addition to that, a secondary spindle on TCs for the default spindles also belongs to the station.

Use the copy function to select a spindle.

1. Mark the spindle you want to copy in the **Spindle** list.
2. Mark the cells in the mode table you want the name copied to.
3. Press the << button next to the Spindle list.

To clear the selected Spindle from a number of cells mark them and press the **Clear** button next to the Spindle list.

If a spindle is not explicitly selected the default spindle of the bolt will be used (set up using System /Bolt node). This means that if the default spindle is changed for the bolt, the new spindle will automatically be used to tighten the bolt.

Press **Print** to print the current Mode Table configuration. The document printed on your systems default printer includes the settings for all used modes, that is, all modes for which a name is specified.

6.5 Tightening Program

The Tightening Program defines the algorithm to use for tightening a bolt. This includes evaluating the result of the tightening, during execution as well as the final result, and fault handling.

A Tightening Program consists of the following parts:

- Common settings like name, program number, etc.
- User variables and their values
- Steps, the basic blocks to build a program with. The parameters of each step is divided in the following parts:
  - Control – Defines function of the step, i.e. when to switch off, in what direction to run, etc.
  - Restrictions – Used for supervision of the control function. All the restriction tests are executed in parallel with the control function and if any limit is exceeded it will interrupt the execution of the step.
  - Checks – Defines variables that are measured during the execution of the step but are not evaluated until the step has finished. Checks never interrupt the execution of the step.
  - Reject Management – Defines how to do an automatic repair in case of a step failure.
  - Speed – Defines which speed to use during the step.
  - Other – Used for setting up start delays, stop conditions, etc.
- Bolt monitoring. Defines how the complete tightening cycle should be evaluated. It is configured on the Checks tab for the CE-steps. Bolt monitoring differs depending on at which CE-step the program execution ends.
- Trace settings

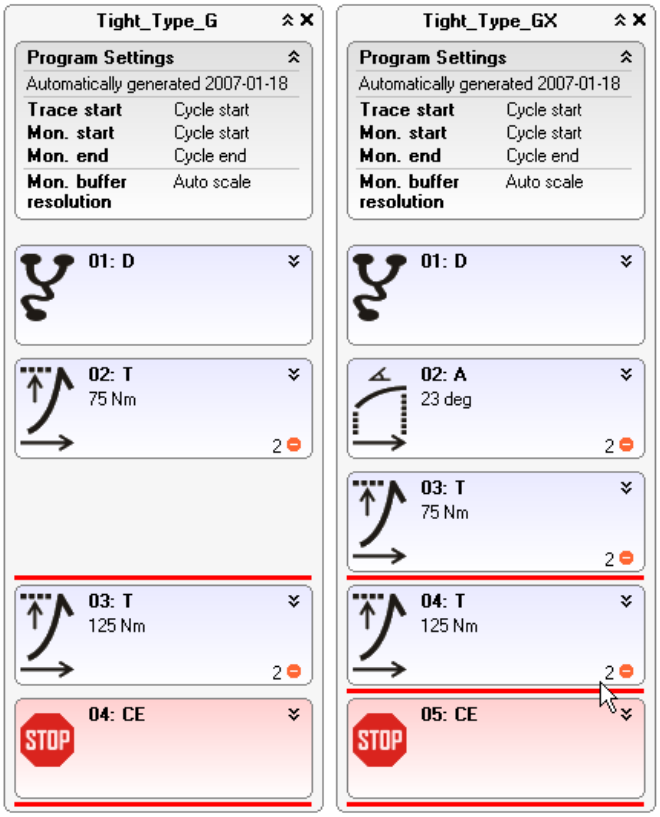
Synchronization

You can choose if the execution of the steps should be synchronized between different spindles or not.

When a station starts a tightening cycle the individual bolts may run freely as long as the steps are not synchronized. When the spindle encounters a synchronized step it will wait for all other spindles to reach their synchronization steps. Spindles may run different number of steps between the synchronization points.

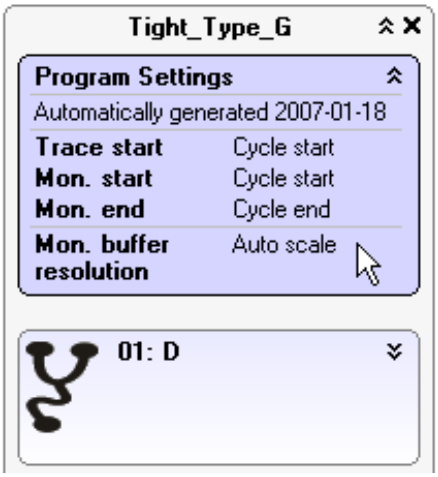
This function is best used in in-homogenous stations with spindles that do individual jobs, e.g. a marriage point in a car assembly.

It is possible to compare how programs will run in relation to eachother by having two or more programs open in the **Tightening Program** form and clicking the **Match Sync. lines** toolbar button.


Tight_Type_G
Program Settings
Automatically generated 2007-01-18
Trace start Cycle start
Mon. start Cycle start
Mon. end Cycle end
Mon. buffer resolution Auto scale
01: D
02: T
75 Nm
2
03: T
125 Nm
2
04: CE
STOP
Tight_Type_GX
Program Settings
Automatically generated 2007-01-18
Trace start Cycle start
Mon. start Cycle start
Mon. end Cycle end
Mon. buffer resolution Auto scale
01: D
02: A
23 deg
2
03: T
75 Nm
2
04: T
125 Nm
2
05: CE
STOP

6.5.1 Program Properties

A tightening program has a number of general properties, the are shown in the **Details** window when the **Program Settings** box is selected.



6.5.1.1 Program Common properties

The most basic settings for a program are located in the **Common** group.

- **Pgm No.:** A tightening program is identified by its name. However, using this property you may optionally specify a Program number as an alternative identifier. This number can be used for reporting and is mandatory when program from PLC is used, as the PLC configures which program to run using the **Pgm. No**.
- **Program Name:** The name of the program.
- **Comment:** Any comment about the program.

6.5.1.2 Program User variables

This group contains the defined user variables for a program.

- **Name:** The name of the variable.
- **Description:** Variable description.
- **Value:** The value of the variable when the program is run.
- **Unit:** The unit of the variable (for information only).

User Variables are used to make programs more flexible and more easy to manage. If a user variable is defined in the program you can later refer to that variable when entering parameters for a step, check, restriction instead of hardcoding actual values.

Tightening

Suppose we want to make two programs, one with target torque 50 Nm and one with target torque 75 Nm and we want to make sure the monitoring Peak Torque check is within 3 % of the target torque in order to have an OK cycle.

Then we can create one program with user defined variables, we call this program **Tighten.** One variable is defined **TargetTorque** and we set this to 50 Nm (our first program).

Then we add a Peak Torque to the CE monitoring and enter the high and low limits using the TargetTorque user variable.

In order to create or second program with target torque 75 Nm we copy the **Tighten** program and change the **TargetTorque** variable value to 75 Nm. The advantage with user variables is that each program need only to expose a set of variables that are sufficient to change to adapt the program to other requirements.

Note! Complex expressions are not supported for user variables, one operation (**+**, **-**, ***, /**) and one parameter is allowed. Examples:

TargetTorque * 1,3
Target1 + 3

6.5.1.3 Program Settings

Trace Settings

- **Trace Start:** When trace collection should start, the options are Cycle Start or at a specific step. In many cases you are not interested in the first part of the tightening, but want more details in the latter parts
- **Trace Start Quantity:** Makes it possible to start trace collection when a specific condition has ocurred, for example a torque value has been reached.
- **Include 2nd Torque:** Check this to include the second torque channel in the trace if equipped (takes more memory).

- **Include 2nd Angle:** Check this to include the second angle channel in the trace if equipped (takes more memory).

Monitoring Settings

- **Stop monitoring when NOK:** Check this checkbox to have the monitoring function stop recording data whenever a step is found NOK when evaluated. This, together with the step control parameter **Start/restart monitoring** (see chapter: Step – Control), is used to select which parts of the cycle that is evaluated by he monitoring checks set up using the Cycle End step(s). See also chapter: Bolt Monitoring.
- **Use 2nd monitoring buffer:** Check this checkbox to enable the second monitoring buffer. The buffer will use the same settings as the monitoring buffer. Only Final torque and Threshold angle checks are run and reported for the second buffer. See also chapter: Bolt Monitoring.
- **Monitoring Start and Monitoring End:** Are used to specify the condition when monitoring function should be started and stopped.
- **Auto scale Mon Buffer:** If Auto scale is checked then the buffer is automatically rescaled if the traveled angle exceeds the maximum angle of the buffer. When started every element covers 1 encoder increment and whenever the buffer limit is reached this value is doubled causing a lower resolution. Already recorded data is kept (compressed) when the buffer is rescaled. If you leave Auto scale unchecked you must manually enter the resolution of the buffer in the edit field to the right of it. In this case the oldest recorded data will be overwritten when buffer limit is reached resulting in that the last part of the cycle is kept in the buffer. When using a fix buffer the checkbox **Allow monitoring buffer overrun** controls whether or not a buffer overrun should be regarded as an error or not. If checked then an overrun will generate an event and cause the cycle to end NOK. If not checked then only a warning is generated.

Limits

- **Report threshold torque limit:** Each bolt for which the highest measured torque does not exceeds the **Report threshold torque limit** will have the "Warnings" bit REPLIMIT (REPort LIMIT) set.

  The highest measured torque is defined as the highest value of the two result variables “Bolt Max T” and “Bolt Min T”. If “Bolt Min T” is negative the absolute value of “Bolt Min T” is used instead, that is,. the warning bit is set if *Report threshold torque limit > MAX (ABS (Max T), ABS (Min T))*.

  If none of the bolts executed for a cycle (disconnected bolts excluded) exceeded the value of the **Report threshold torque limit** parameter then the corresponding cycle data will be automatically dropped.

  Normally this parameter is used to avoid reporting tightening made "in the air". Leave the parameter empty to disable the function.
- **Remove fastener torque limit:** This is a "program global" restriction, that is, a restriction that is checked over all steps executed in the cycle. When the measured torque exceeds the entered value the current step is immediately stopped.

An event and the error bit RFTLIMRE (Remove Fastener Torque Limit REeached) indicate the condition.

Backlash compensation (only available for Gauging)

All steps will do backlash compensation, if the direction is different from the previous step and a measured backlash correction exists (measured with step BCT – Backlash correction (only available for Gauging)

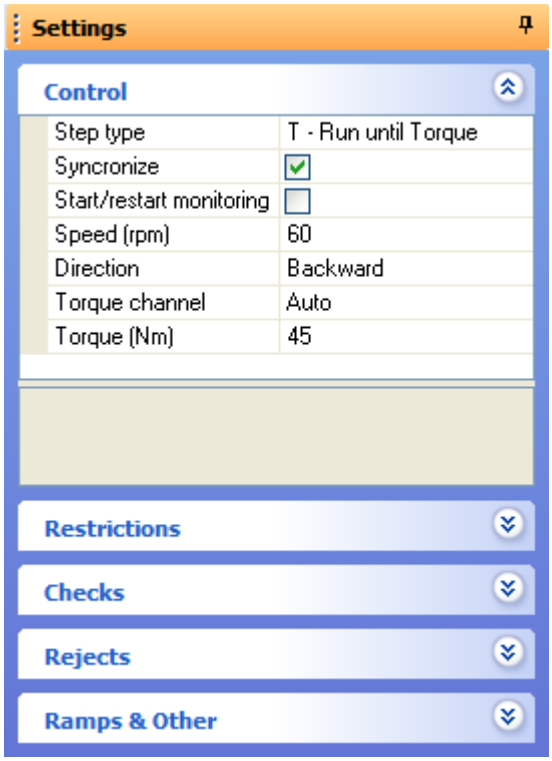
During the backlash correction nothing in the steps will be run. This means that all checks will start after the backlash correction is finished. The same is true for the step. They will not start to execute the control function before the backlash is finished. This means that a step will not end even if the target torque is reached during the backlash. It also means that all angle measurements will start when the backlash is finished.

- **Reset backlash**: If checked the stored backlash value (from an earlier cycle) will be reset at cycle start.
  Default value is checked
- **Speed**: The speed used for the backlash compensation. All steps in the program will use the same speed.
- **Angle channel**: Channel for **Backlash** and **Position** measurements. The backlash compensation and the position measurements are made on the same angle channel.

The stored backlash compensation and positions (**Null, Half, Peak** and **Low**) will be cleared on cycle start if the program uses a different channel compared to the previous program that was run.

6.5.2 Step – Control

The *Control part* describes what main task to perform during the step.



The settings available for the Control part depends on the selected step. It is possible to change step by the parameter **Step type**.

The checkbox **Synchronize** is used to select if the step should synchronize or not.

The check box **Start/restart monitoring** can be used in conjunction with the monitoring settings for the program (Program Settings) in order to start or restart the recording into the monitor buffer. If restarted everything recorded before this step is erased. See also the description of Bolt Monitoring.

Most step that run the spindle have a **Speed** parameter and a step **Direction** that can be set, note that the direction is relative to the desired spindle direction.

It is also possible to select the torque and angle channel to use for step control with the **Torque channel** and **Angle channel** properties. The default value is *Auto* which is the first channel (Torque 1 or Angle 1) if not disabled on the Spindle Set Up form.

Below the common properties follows the step specific properties, the following sub chapters list all available step types and describe the parameters.

6.5.2.1 DT - Run until Dyna-Tork™

Step type Id: 1

Parameters: T (Torque), TI1 (Time1), TI2 (Time2), PERC (Current percentage), DIR (Direction).

Function: Run spindle, in DIR direction, until torque is T using current control. When the target is reached a new torque, DT, is calculated as

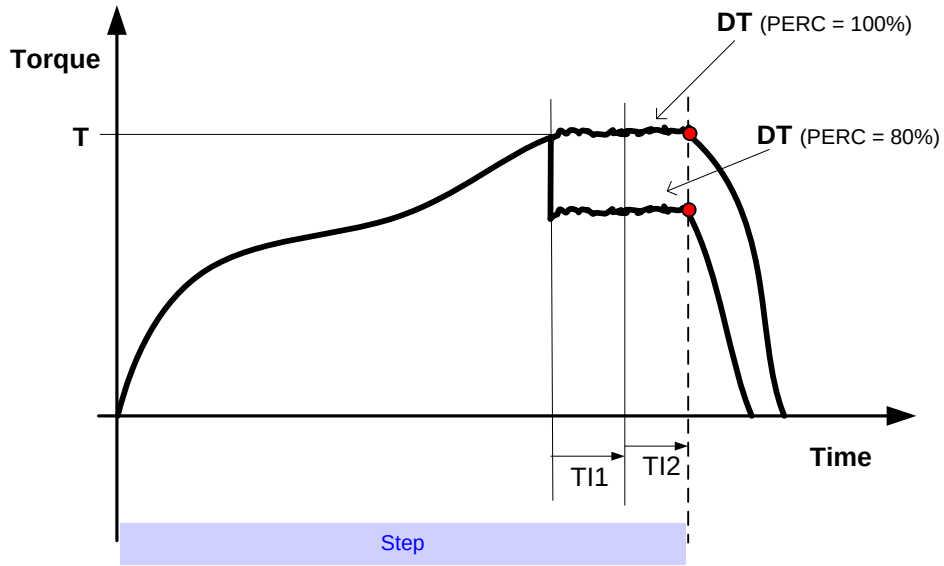
DT = T * PERC / 100

The torque DT is then retained during the next TI1 + TI2 seconds using current control.

The PERC parameter makes it is possible to Dyna-Tork™ at a lower level than the target.

Before the step is started both torques, T and DT, are converted to corresponding motor currents using the spindle parameters **T/C Factor, T/C Correction Factor** and **Gear ratio**. When the step starts the current that corresponds to T is set as current reference to the motor servo. When the servo reaches this current the step immediately switches the current reference to the current that corresponds to DT which then is maintained for TI1 + TI2 seconds.

Diagram:



T	Torque	**TI1**	DynaTork Time 1
PERC	Percent of T	**TI2**	DynaTork Time 2
DT	Torque level retained during TI1 and TI2		

Result var.: This step produces the step level result variables “DT T”, "DT Mean T" and "Relax A".

"DT T" is the torque measured when the torque set point is switched to DT.

"DT Mean T" is the mean torque measured during the second time interval (TI2).

"Relax A" (relaxation angle) is the angle that the bolt moves during the time interval TI1 + TI2.

Trace: Torque T is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

6.5.2.2 DT2 - Run until Dyna-Tork™, method 2

Step type Id: 34

Parameters: T (Torque), TI1 (Time1), TI2 (Time2), PERC (Current percentage), DIR (Direction).

Function: Run spindle, in DIR direction, until torque is T using transducer control. When the target is reached the current measured on the current input channel at this point is sampled and a new current set point is calculated as

Current Set point = Measured current * PERC / 100

The current is then retained during the next TI1 + TI2 seconds using current control.

The PERC parameter makes it is possible to Dyna-Tork™ at a lower level than the target.

Result var.: This step produces the step level result variables “DT T”, "DT Mean T" and "Relax A".

"DT T" is the torque measured when the torque set point is switched to DT.

"DT Mean T" is the mean torque measured during the second time interval (TI2).

"Relax A" (relaxation angle) is the angle that the bolt moves during the time interval TI1 + TI2

Trace: Torque T is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

6.5.2.3 DT3 - Run until Dyna-Tork™, method 3

Step type Id: 35

Parameters: T (Torque), TI1 (Time1), TI2 (Time2), PERC (Current percentage), DIR (Direction).

Function: Run spindle, in DIR direction, until torque is T using transducer control. When the target is reached a new torque, DT, is calculated as

DT = T * PERC / 100

The torque DT is then retained during the next TI1 + TI2 seconds using current control.

The PERC parameter makes it is possible to Dyna-Tork™ at a lower level than the target.

Before the step starts the torque DT is converted to a corresponding motor current using the spindle parameters **T/C Factor, T/C Correction Factor** and **Gear ratio**. The first part of the step runs using transducer control. When the torque measured on the torque transducer used for control reaches T the step switches to current control using the current that corresponds to DT as set point for the motor servo. This current is then maintained for TI1 + TI2 seconds.

Result var.: This step produces the step level result variables “DT T”, "DT Mean T" and "Relax A".

"DT T" is the torque measured when the torque set point is switched to DT.

"DT Mean T" is the mean torque measured during the second time interval (TI2).

"Relax A" (relaxation angle) is the angle that the bolt moves during the time interval TI1 + TI2.

Trace: Torque T is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

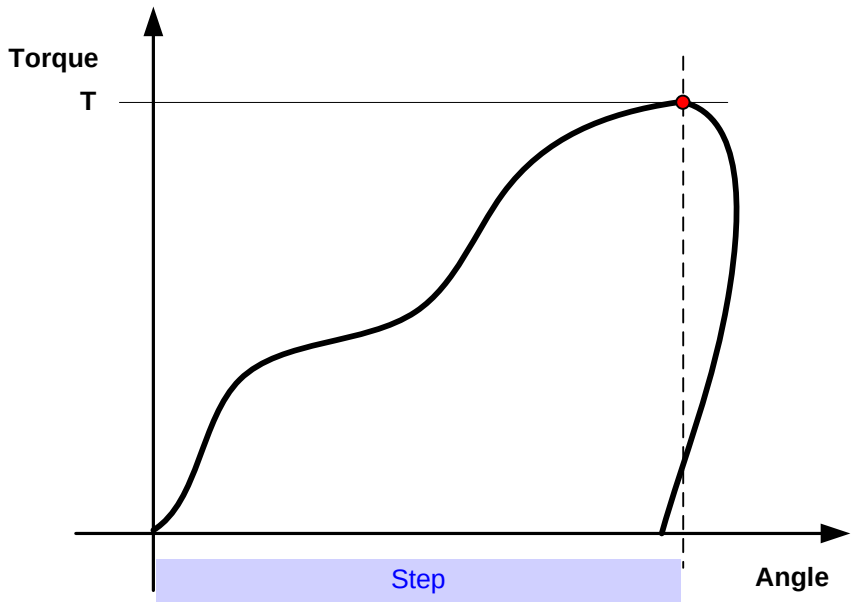
6.5.2.4 T - Run until Torque

Step type Id: 2

Parameters: T (Torque), DIR (Direction).

Function: Run spindle, in DIR direction, until torque is T.

Diagram:



T	Target Torque

Trace: Torque T is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

6.5.2.5 TC - Run until Torque with Current Control

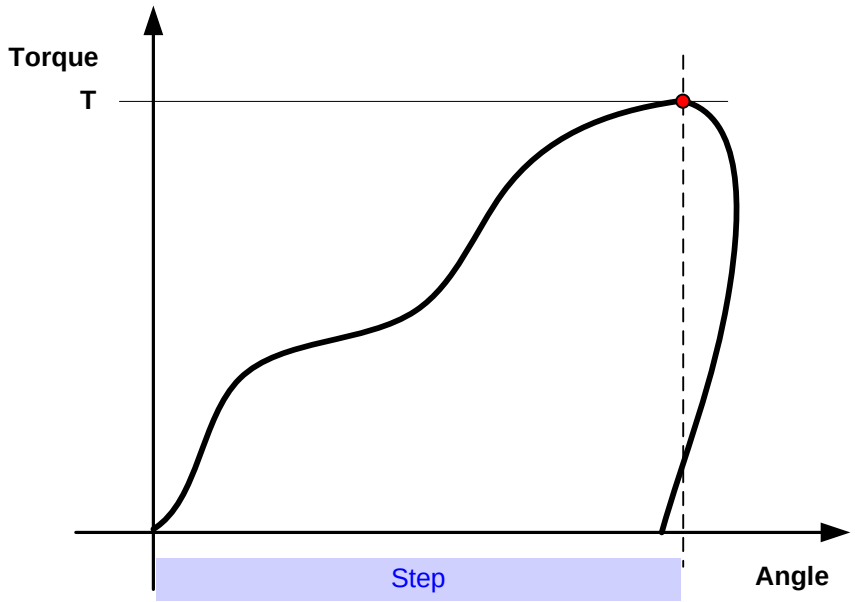
Step type Id: 3

Parameters: T (Torque), DIR (Direction).

Function: Run spindle, in DIR direction, until torque is T using current control.

The target torque T is converted to a corresponding motor current using the spindle parameters **T/C Factor, T/C Correction Factor** and **Gear ratio**. This current is then set as reference to the motor servo. The step is stopped when the servo reaches its set point.

Diagram:



T	Target Torque

Trace: Torque T is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

6.5.2.6 AO - Run until Angle with overshoot compensation

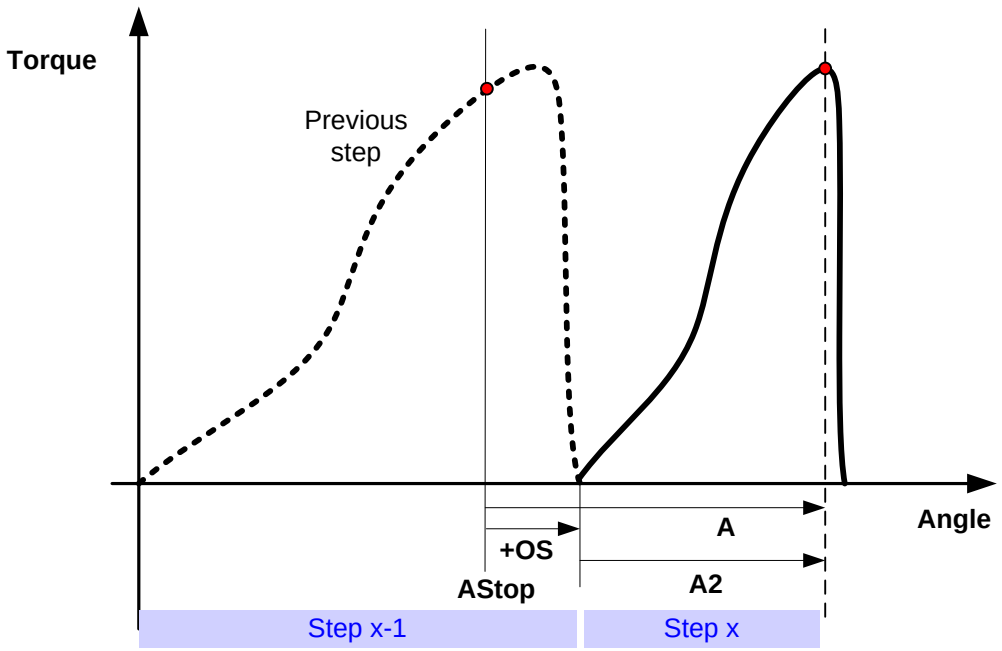
Step type Id: 5

Parameters: A (Angle), DIR (Direction).

Function: Run spindle, in DIR direction, an angle of A degrees. The angle is measured from where the previous step was stopped, that is from its shut off.

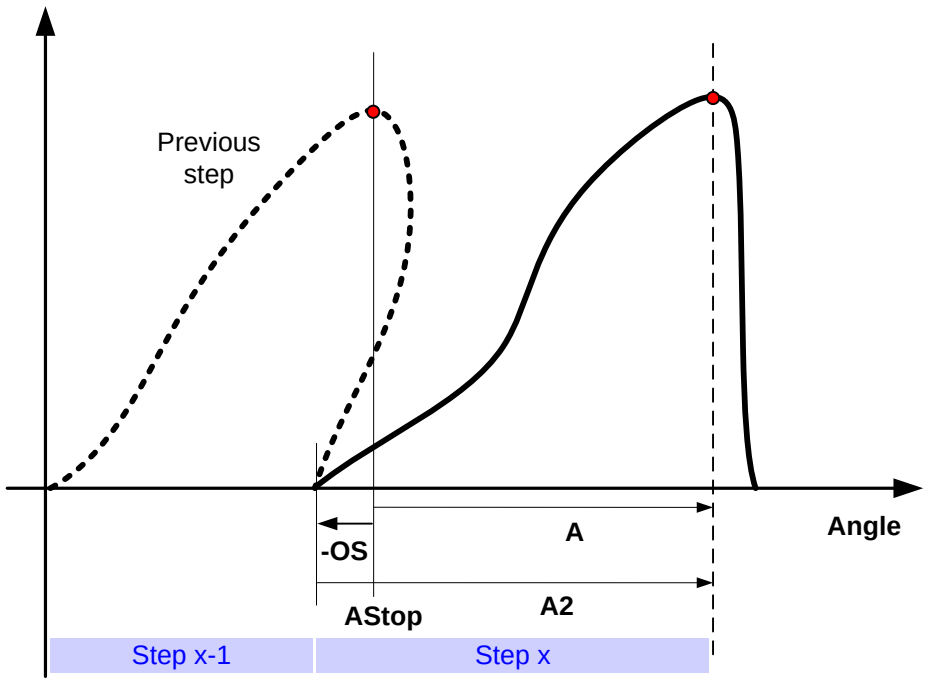
Any overshoot will be included in this step. If the overshoot is larger than A the step will be signaled ready immediately. Overshoot is defined as the angle measured between shut off point in the previous step and the current angle when this step starts.

Diagram 1:



Astop	Shut-off point	**A**	Target Angle
A2	A - OS actual angle movement in this step		
OS	Overshoot in previous step		

Diagram 2:



Astop	Shut-off point	**A**	Target Angle
A2	A - OS actual angle movement in this step		
OS	Overshoot in previous step		

Trace: Angle A is shown as a horizontal line from step start to step stop in trace when Angle vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

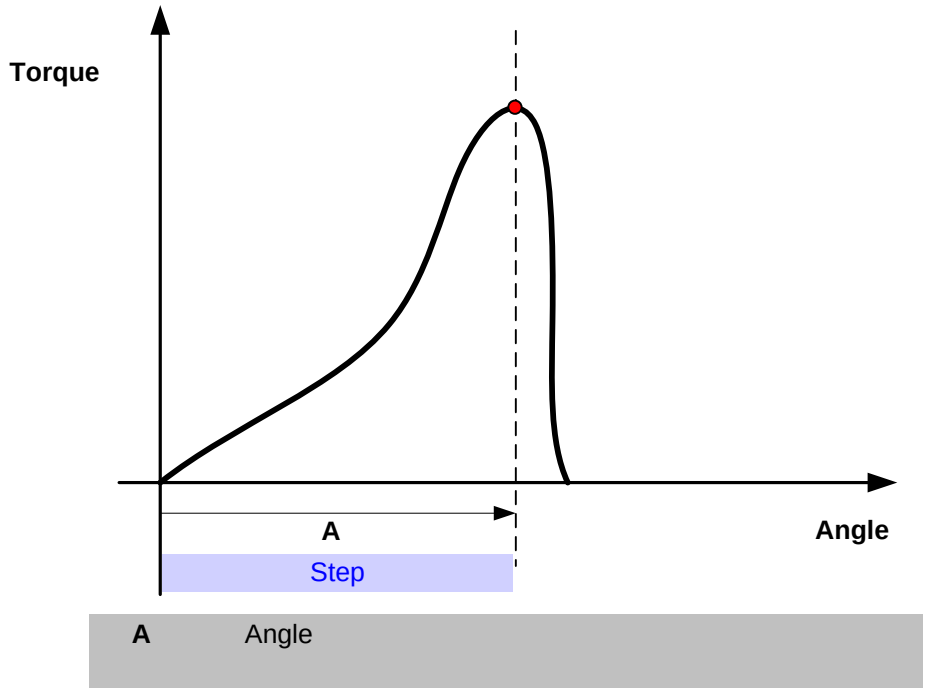
6.5.2.7 A - Run until Angle

Step type Id: 6

Parameters: A (Angle), DIR (Direction), Start condition.

Function: Run spindle, in DIR direction, until angle A has passed.

Diagram :



Trace: Angle A is shown as a horizontal line from step start to step stop in trace when Angle vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

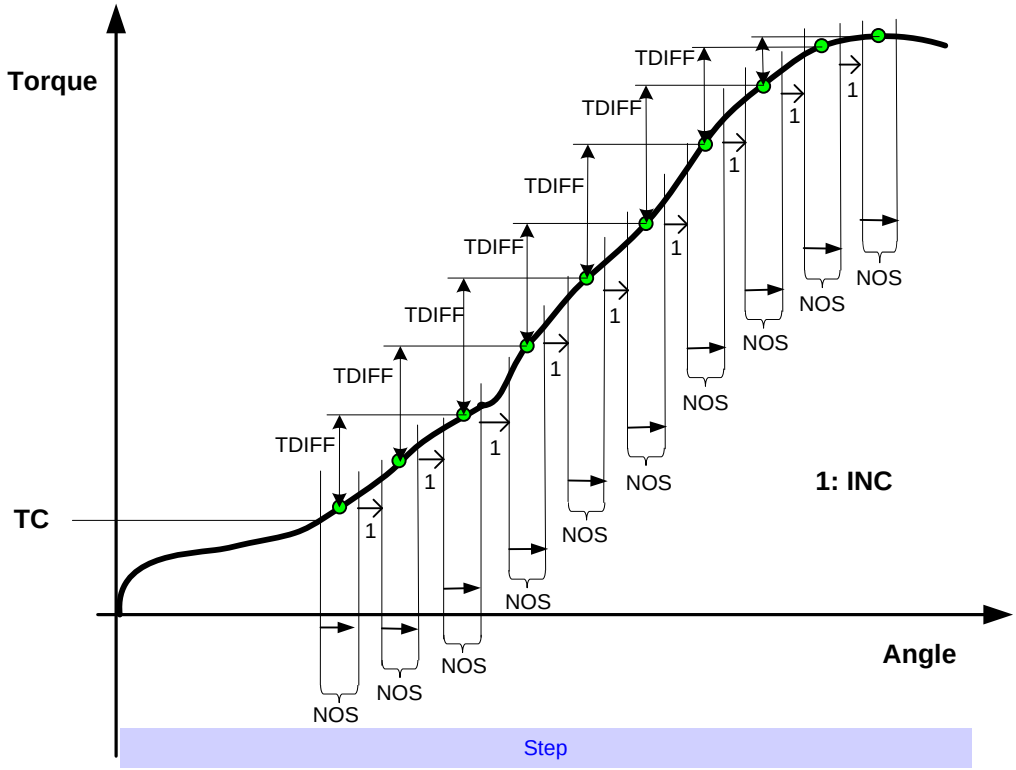
6.5.2.8 Y1 - Run until Yield point, method 1

Step type Id: 7

Parameters: TC (Torque TC), INC (Increment), NOS (No degrees), TDIFF (Torque difference), DIR (Direction).

Function: Run spindle, in DIR direction, until yield point. Search for yield point starts when the torque has reached TC. The average of the torque (T_n) over NOS angle degrees is calculated. This procedure is repeated after INC angle degrees. The difference in average torque values (T_n - T_{n-2}) are continuously calculated and the max difference is stored.
The yield point is considered reached when the calculated difference is less than TDIFF % of the current max difference.

Diagram:



TC	Torque TC	**INC**	Increment
NOS	No. degrees	**TDIFF**	Difference in average torque

Trace: Torque TC is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

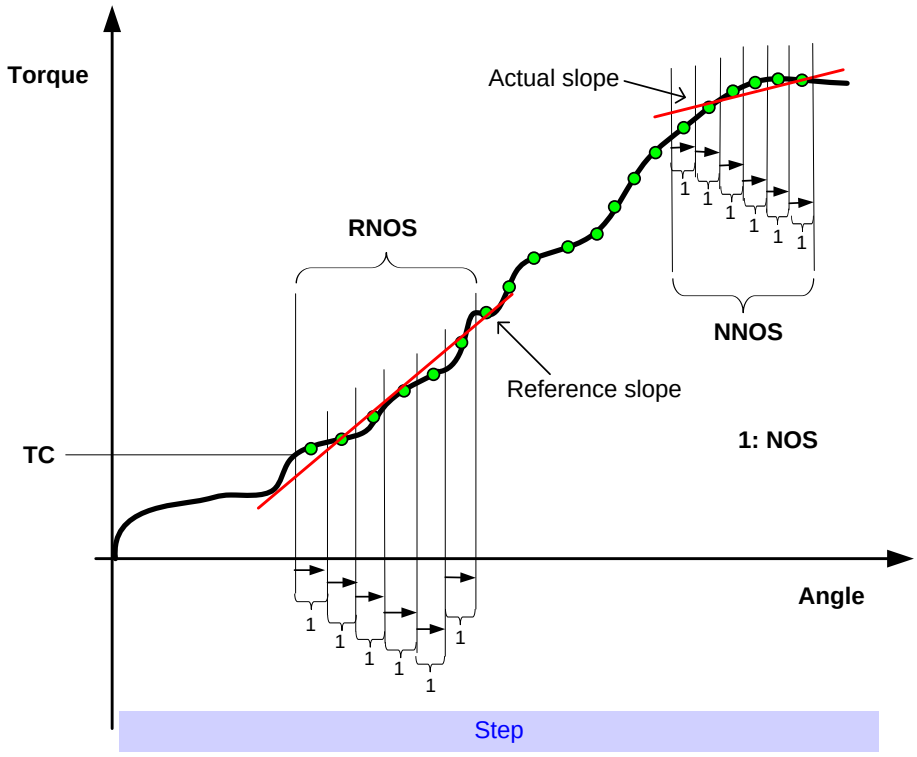
6.5.2.9 Y2 - Run until Yield point, method 2

Step type Id: 8

Parameters: TC (Torque TC), NOS (No degrees), NNOS (No of points), TDIFF (Torque difference), RNOS (No of points ref), DIR (Direction).

Function: Run spindle, in DIR direction, until the yield point. Search for the yield point starts when the torque has reached TC. The average of the torque over NOS angle degrees is calculated. This procedure is repeated RNOS times. A reference slope is calculated with linear regression over the RNOS points. After that, new average values are continuously calculated over NOS degrees. The actual slope is calculated with linear regression over the last NNOS average points. The yield point is reached when the actual slope is less than TDIFF % of the reference slope.

Diagram:



TC	Torque, TC	**NNOS**	**TDIFF**
NOS	No. degrees	**RNOS**	

Trace: Torque TC is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

6.5.2.10 LT - Loosen until torque

Step type Id: 11

Parameters: T (Torque), Mask1, Mask2, DIR (Direction).

Function: Run until torque decreases below the threshold level T.

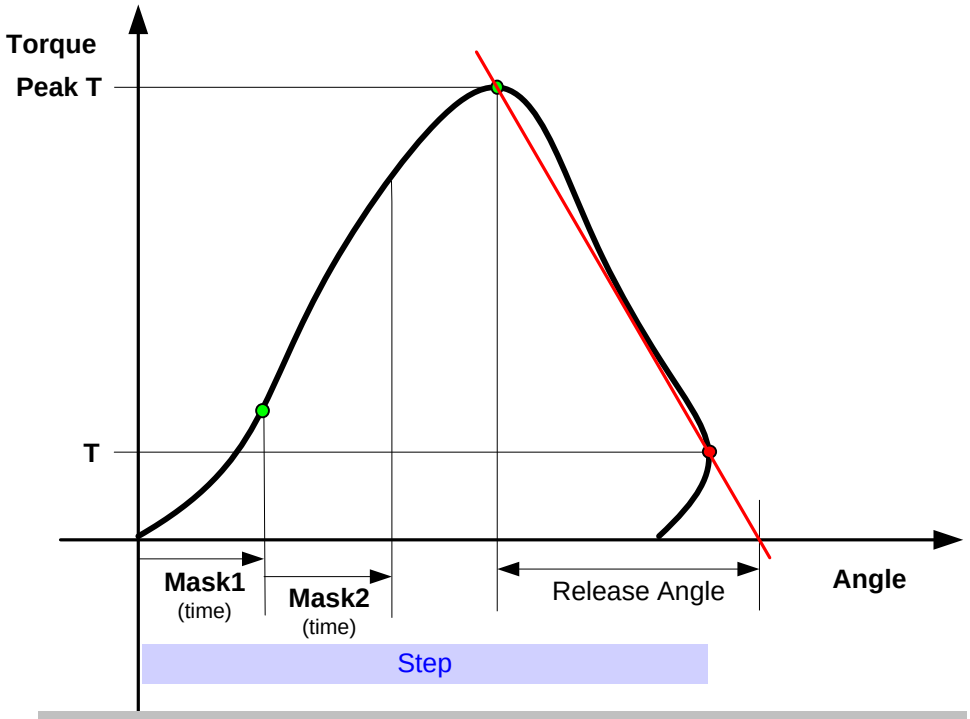
First run for Mask1 seconds in the DIR direction, then measure the torque. Should the measured torque be less than T the step is stopped with status OK.

Continue to run for Mask2 seconds in the DIR direction. During this time the step is not shut off regardless of the measured torque.

From this point continue to run in the DIR direction until the measured torque is less than, or equal to, the parameter T.

The Release angle can be reported from this step. It is measured from the point where the torque reaches its highest value to the point where the projected torque vs. angle curve should have crossed the angle axis if not shut off. Please note that the peak torque is not checked for during Mask1.

Diagram:



T	Torque	**Peak T**	Peak Torque
Mask1	Time 1		
Mask2	Time 2		

Trace: Torque T is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

6.5.2.11 RA - Release angle

Step type Id: 12

Parameters: T (Torque), Mask1, Mask2, DIR (Direction).

Function: Run until torque decreases below 0.3 * peak torque (measured during the step).

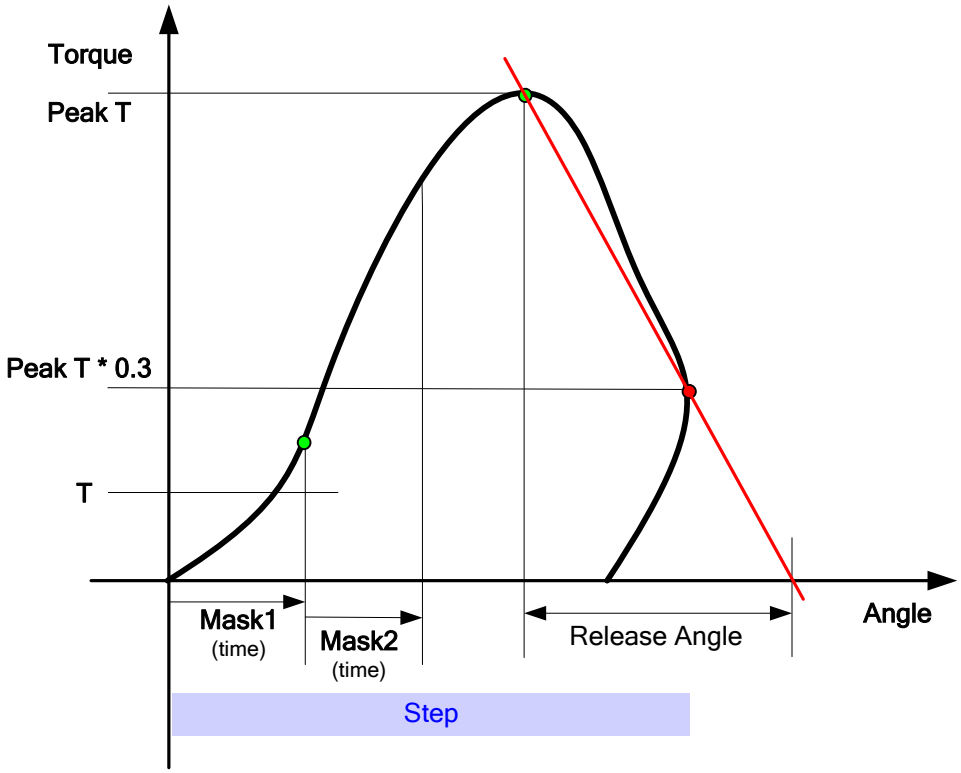
First run for Mask1 seconds in the DIR direction then measure the torque. Should the measured torque be less than T the step is stopped with status OK.

Start monitoring of peak torque and continue to run for Mask2 seconds in the DIR direction. During this time the step is not shut off regardless of the measured torque.

From this point continue to run in the DIR direction until the measured torque is less than 0.3 * the peak torque (measured from that the timer Mask2 was started).

The Release angle can be reported from this step. It is measured from the point where the torque reaches its highest value to the point where the projected torque vs. angle curve should have crossed the angle axis if not shut off. Please note that the peak torque is not checked for during Mask1.

Diagram:



T	Torque	**Peak T**	
Mask1	Time 1	**Peak T * 0.3**	Shut off point
Mask2	Time 2		

Trace: Torque T is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

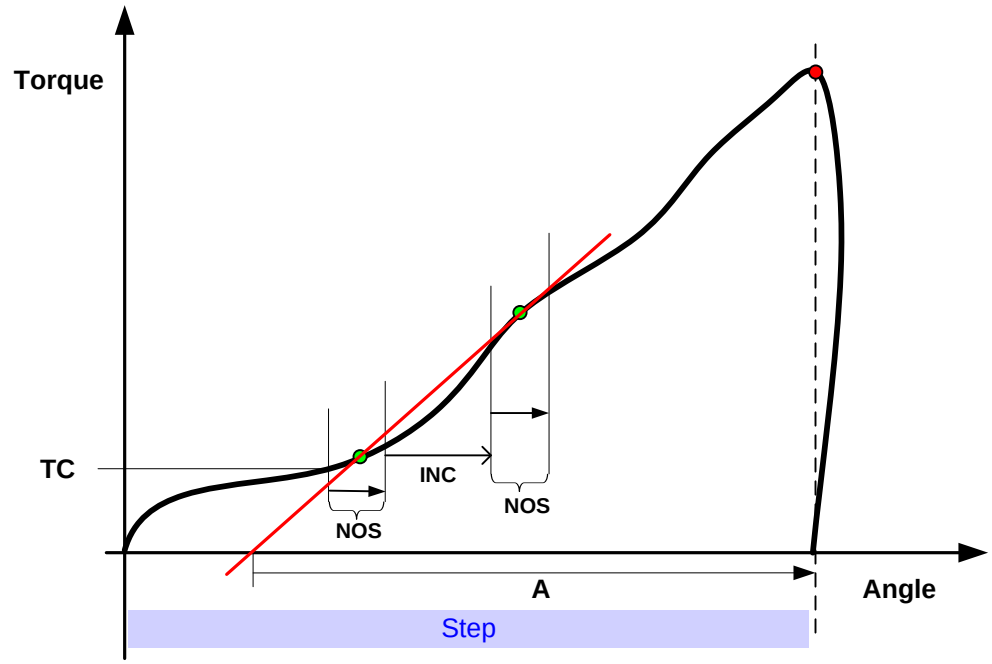
6.5.2.12 PA - Run until Projected angle

Step type Id: 9

Parameters: A (Angle), TC (Torque, TC), INC (Increment), NOS (No degrees), DIR (Direction).

Function: Run spindle, in DIR direction, until projected angle A. Search starts when torque has reached TC. The average of the torque over NOS angle degrees is calculated. This procedure is repeated after INC angle degrees. A straight line is calculated through the two calculated torque average values and down to the x-axis. The projected angle starts from this intersection.

Diagram:



TC	Torque	**INC**	Increment
NOS	No. degrees	**A**	Projected Angle

Trace: Torque TC is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

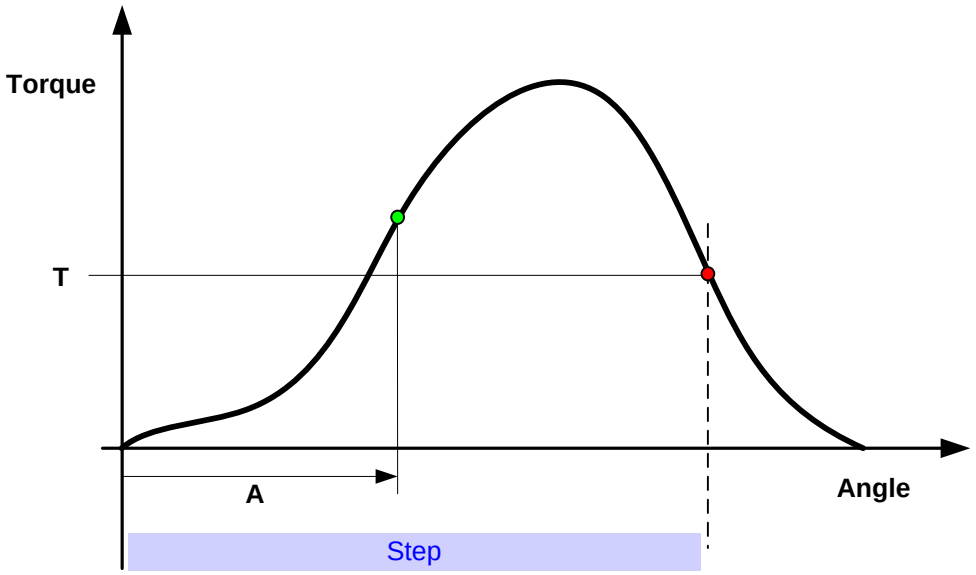
6.5.2.13 TD - Run until Torque has decreased

Step type Id: 10

Parameters: T (Torque), A (Angle), DIR (Direction)

Function: Run spindle, in DIR direction, until angle A is reached. The rotation continues but from that point the torque is measured. The step stops when the measured torque is less than, or equal to, T.

Diagram:



A	Angle
T	Torque

Trace: Torque T is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

6.5.2.14 TI - Run until Time

Step type Id: 13

Parameters: TI (Time), DIR (Direction).

Function: Run spindle, in DIR direction, for TI seconds.

Trace: Time TI is shown as a vertical line in trace when Torque, Angle, Current or Current as torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

6.5.2.15 E1 - Run Engage method 1

Step type Id: 14

Parameters: TI (Time), T (Torque), TC (Current as Torque).

Function: Engage spindle, method 1. Run spindle reverse for TI seconds and then forward for TI seconds or until torque T or current TC is reached.

Trace: Torque T is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Current C is shown as a horizontal line from step start to step stop in trace when Current vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

6.5.2.16 E2 - Run Engage method 2

Step type Id: 15

Parameters: TC (Current As Torque), TI (Time), DIR (Direction).

Function: Engage spindle, method 2. Run spindle, in DIR direction, with maximum TC torque, until no increase in angle for TI seconds.

Trace: TC is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

6.5.2.17 NS - Run until next step

Step type Id: 17

Parameters: DIR (Direction).

Function: Run spindle, in DIR direction, until next step starts. Next step starts when all the other spindles reach their next synchronization point. That is, this step is meaningful only if there are other spindles executing normal steps at the same time.

Trace: There are no control parameters for this step type in trace.

6.5.2.18 GS - Run Gear Shift

Step type Id: 16

Parameters: GS (Gear Shift), TI1 (Time1), TI2 (Time2).

Function: Controls the digital output used for shifting the gear for the currently executed spindle. GS equal to 1 (one) will cause gearshift output go high when this step is executed. GS equal to 0 (zero) will make the gearshift output go low.

When changing speed from high to low (normal tightening), the spindle will first run reverse for TI1 seconds, then wait for TI2 seconds before the gear is changed.

When changing speed from low to high the spindle will first run forward for TI1 seconds, and then wait for TI2 seconds before the gear is changed.

Trace: There are no control parameters for this step type in trace.

6.5.2.19 PLC - Run PLC step

Step type Id: 18

Parameters: DIR, PLC output STEPTYPE, PLC output STEPVAL

Function: This step is configurable during run-time using the PLC. When the PLC sets the start signal high the PLC Station variables STEPTYPE and STEPVAL are sampled. The first output, STEPTYPE, controls the type of step that will be executed and the STEPVAL is used as target value. DIR is the only parameter that is defined in the tightening program and it controls the direction of the step.

STEPTYPE can take the following values:

STEPTYPE	Type of Step
1	T - Run until Torque
2	A - Run until Angle with Start condition = AS1
3	TI - Run until Time

STEPVAL should be set to a target value of corresponding type, that is, a torque, an angle, or a time.

6.5.2.20 JOG - Run until digital input goes high / low

Step type Id: 19

Parameters: Stop Condition, DIR (Direction), Digital Input.

Function: This step runs the current spindle, in the direction defined by parameter **DIR**, until the input specified by **Digtal Input** goes high or low depending on the value of **Stop Condition**.

The parameter **Stop Condition** can take the following values:

Stop Condition	Function
FE	Falling Edge – The step runs until a falling edge is detected after step start.
RE	Rising Edge – The step runs until a rising edge is detected after step start.
AE	Any Edge – The step runs until a falling or a rising edge is detected after step start.
Low	The step runs until low value is detected. Should the digital input be low at step start the step is finished immediately.
High	The step runs until high value is detected. Should the digital input be high at step start the step is finished immediately.

The parameter **Digital Input** can take the following values:

Digital Input	Digital input monitored
DI def. by Spindle	The digital input specified by the parameter **Dig input for JOG steps** on the Spindle setup form (see Spindle Setup Application page).
PLC_DI_1, … PLC_DI_10	Can be accessed in the PLC
Local DI 1	One of the four local digital inputs available on TCs of type 'PTC', that is, a System or Station TC.
Local DI 2	
Local DI 3	
Local DI 4	

Trace: There are no control parameters for this step type in trace.

6.5.2.21 W - Wait

Step type Id: 20

Parameters: **Type** (Time, PLC, Digital Input, Continue)
Time, **Digital Input, Stop Condition, Hold Position**

Trace: Not available

Function: Waits until the specified stop condition is fulfilled. Does not run the spindle while waiting.

Type	Parameter	Function
Time	**Time**	The step waits the specified time
PLC		The step waits until a positive edge of one of the PLC Station Variables START or INTERMEDSTART is detected
Digital Input	**Digital input**. Local DI 1, … Local DI 4 PLC_DI_1, … PLC_DI_10 DI Spindle def. **Stop Condition**: Falling Edge, Rising Edge, Any Edge, Low or High	The step waits until the selected digital signal fulfils the condition selected in **Stop Condition**
Continue		The step waits until all other bolts running in parallel have reached the next synchronization point.

If the check box **Hold Position** is checked, the spindle tries to hold the position of the bolt during the wait. The position will be held until the next step starts, if there are any delays between the steps due to synchronization the position will still be kept.

6.5.2.22 TA – Run Torque-Angle with no stop

Step type Id: 28

Parameters: **Torque TC, Angle, TA step type, Falling edge, A start cond, Angle torque start, DIR** (Direction).

Function: Depending on the value of parameter **TA step type** four different functions are possible.

The TA program is considered to be one step, as there is no step change where the spindles can be synchronized. The Angle parameter may be reported using an angle check.

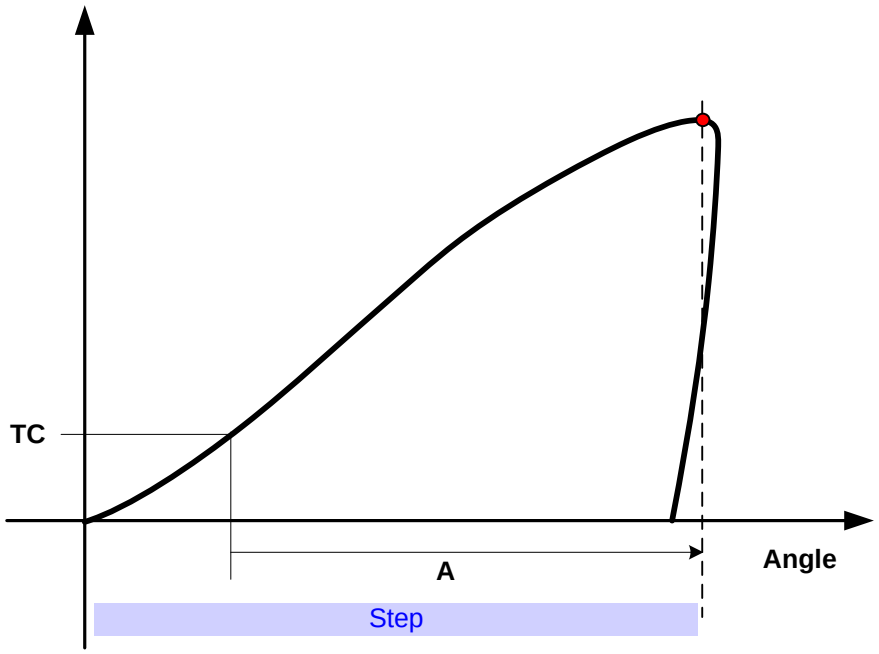
T plus A

Function Run spindle, in DIR direction, until the Torque TC is reached. Measured from this point, continue to run spindle until Angle is reached. The switch from torque to angle control is done "On-The-Fly", that is, without stopping in-between.

The two parts can be programmed with different speeds and if so the speed will be changed on the fly without stopping.

By default the search for the **TC** threshold is done on rising torque.

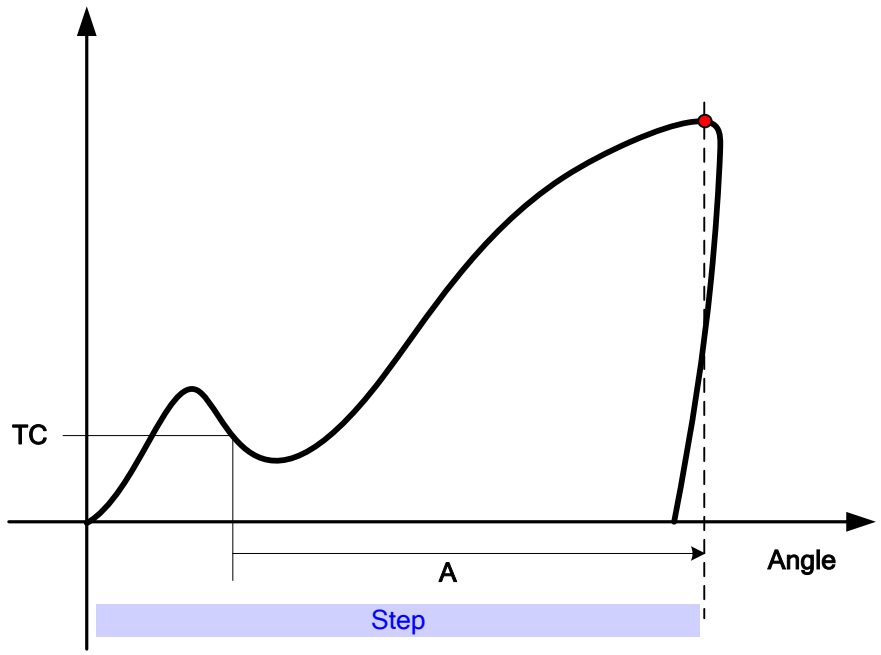
TC on rising edge:



TC	Torque TC
A	Target angle

If parameter **Falling edge** is marked the search for the **TC** threshold is done on falling torque.

TC on falling edge:

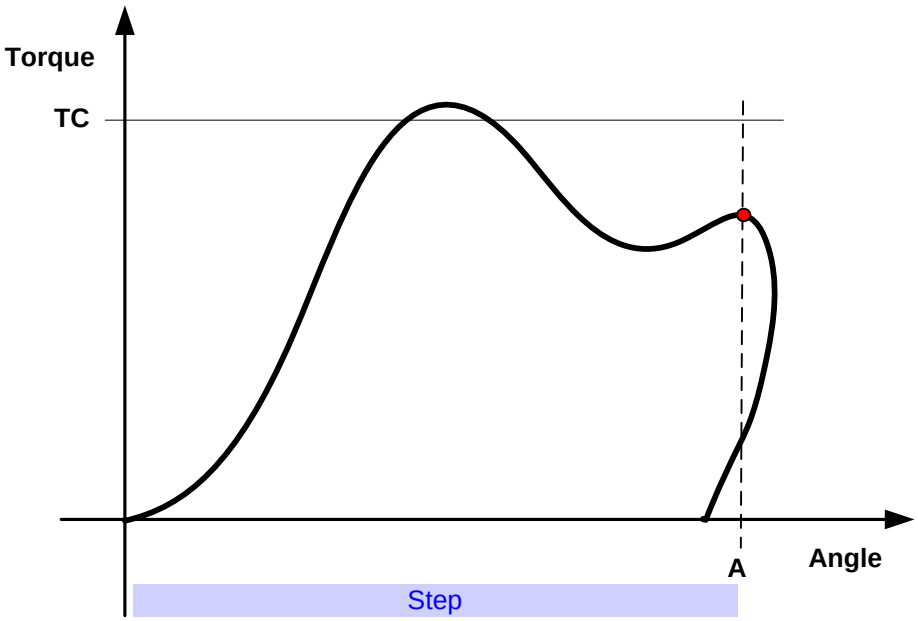


TC	Torque TC
A	Target angle

T and A

Run spindle, in **DIR** direction, until torque has been larger than Torque **TC** and the angle has been larger than **Angle**. Parameter **A start cond**. controls from where **Angle** is measured. See description on following page for possible values.

Diagram:

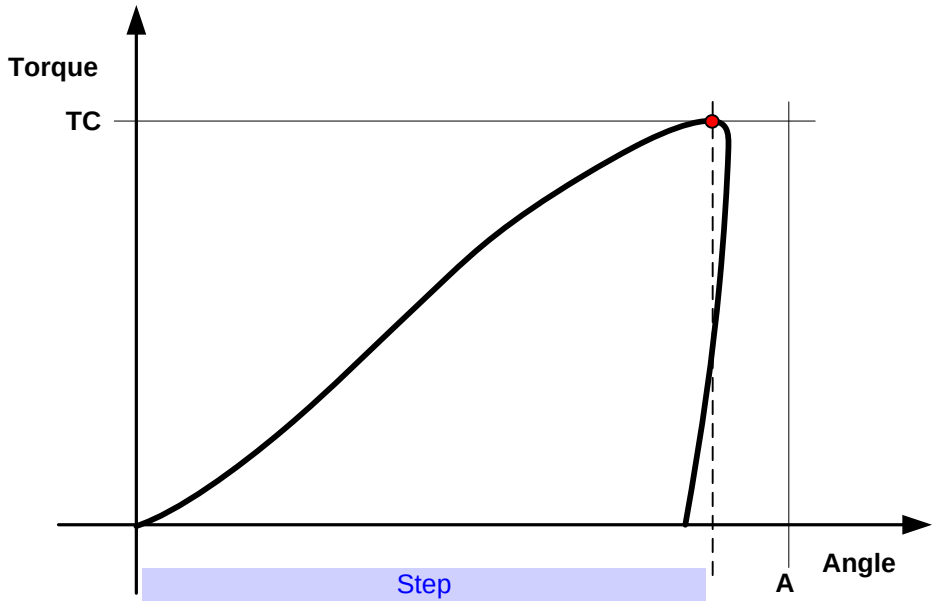


TC	Torque TC
A	Target angle

T or A

Run spindle, in **DIR** direction, until the first of either Torque **TC** or **Angle** is reached. Parameter **A start cond**. controls from where **Angle** is measured. See description on following page for possible values.

Diagram:

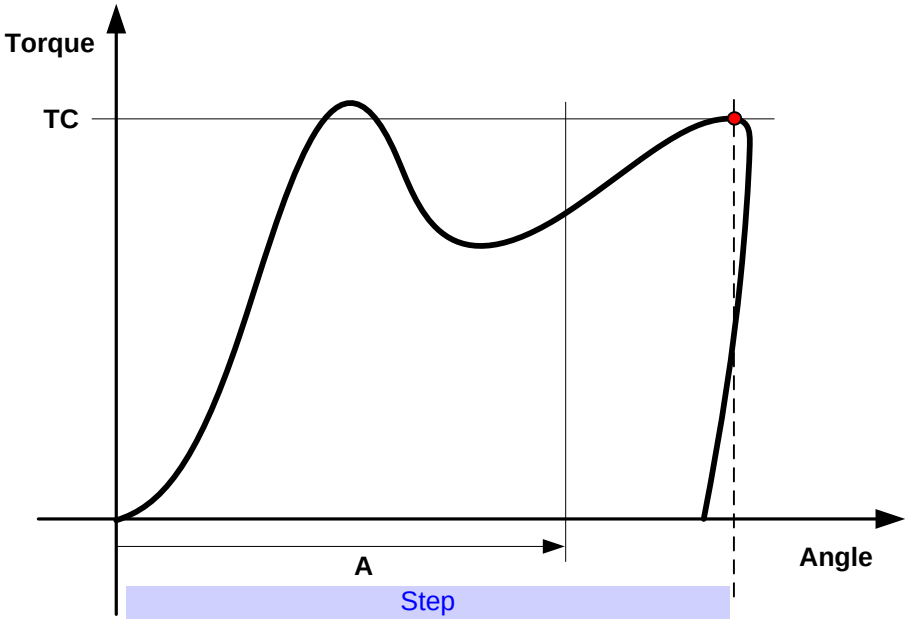


TC	Torque TC
A	Target Angle

A plus T

Run spindle, in **DIR** direction, until the **Angle** is reached. Measured from this point, continue to run spindle until Torque **TC** is reached. The switch from angle to torque control is done "On-The-Fly", that is, without stopping in-between. Parameter **A start cond**. controls from where **Angle** is measured. See description on following page for possible values.

Diagram:



TC	Torque TC
A	Target angle

Parameters **A start cond.** and **Angle torque start** controls from which point the angle Angle is measured. Possible values are:

Start condition	Description
AS1 (Normal)	Angle measurement starts at the angle of the current position when the step is started.
T	Angle measurement starts when the torque passes the level specified using parameter **Angle torque start**.

Trace: Torque **TC** is shown as a horizontal line from step start to step stop in trace when **Torque vs. Time** is selected. Angle **A** is shown as a horizontal line from step start to step stop in trace when **Angle vs. Time** is selected. Control parameters must be enabled in the trace form (see View of trace curves).

6.5.2.23 SR – Socket Release

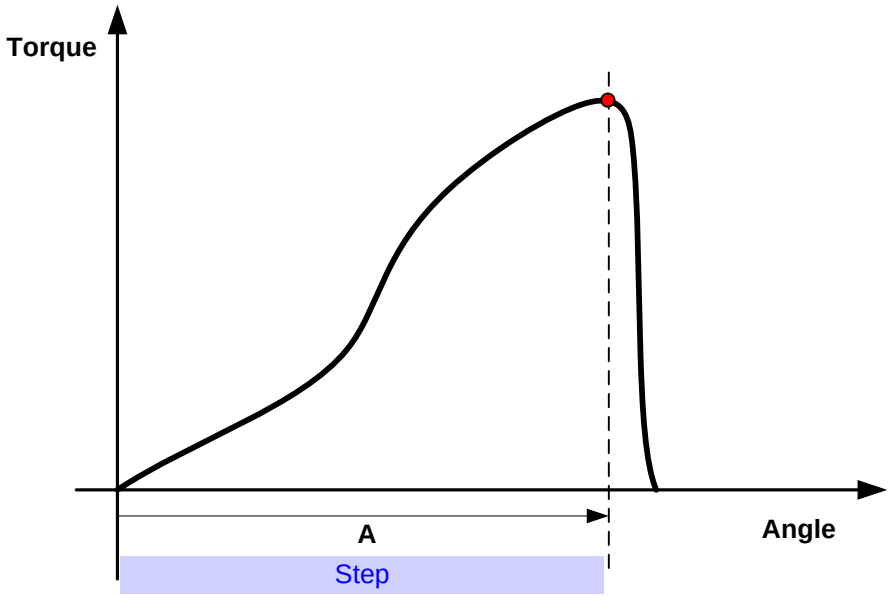
Step type Id: 29

Parameters: A (Angle), DIR (Direction).

Function: Run spindle, in DIR direction, until angle A has passed.. Angle is zero at start of step.

This step performs the same function as a “A - Run until Angle” step with Start condition set to AS1 except from that nothing that this step does is recorded by the Bolt Monitoring function.

Diagram:



A	Target Angle

Trace: Angle A is shown as a horizontal line from step start to step stop in trace when Angle vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

6.5.2.24 ADT-Angle Delta Torque

Step type Id: 31

Parameters: **Delta Torque**, Angle, DIR (Direction).

Function: 1. Run Angle degrees in the reverse direction.
2. Measure the torque when the Angle is reached and store this value as Tlast.
3. Stop the spindle and wait until zero speed is reached.
4. Run in the forward direction until the torque Tlast + Delta Torque is reached.
5. Shut off the step.

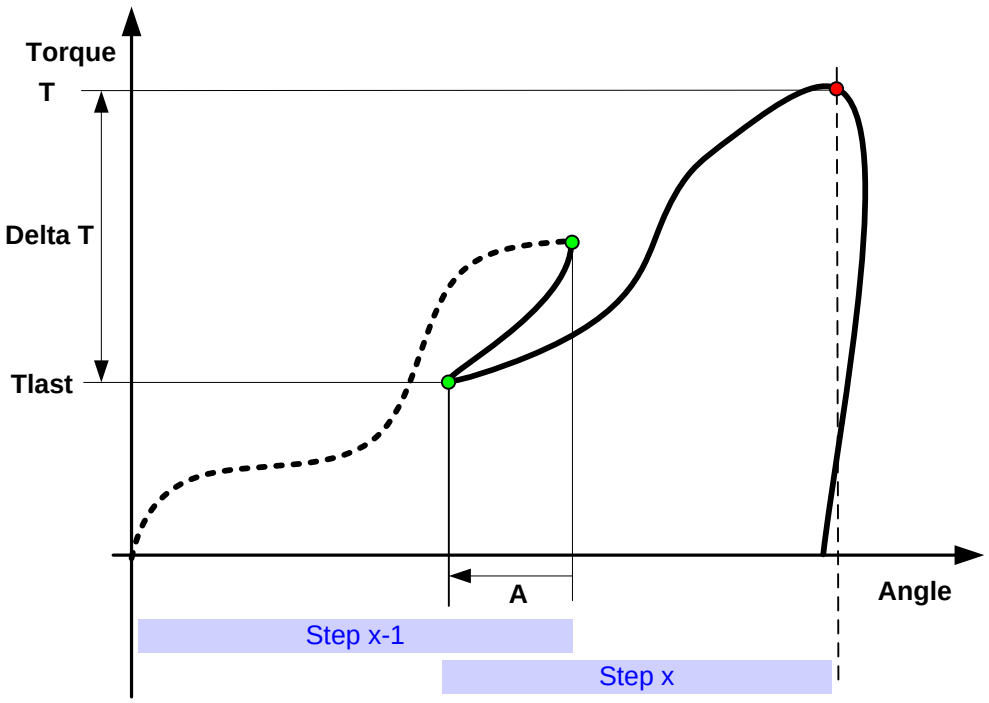
The same speed and speed ramps are used both when running in the backward and the forward directions.

If Check angle is used it will report the total angle without compensation of the lash.

The step is intended to be used in an injector pump setting application and the proposed tightening sequence is:

Step 1: T = 25 [Nm]; Direction = Forward
Step 2: T = 3 [Nm]; Direction = Backward (to deal with the lash in the transmission).
Step 3: ADT = 5 [Nm] / 60 [deg]
Step 4: A = 30 [deg]; Direction = Backward
Step 5: CE

Diagram:



Delta T	Delta Torque
A	Angle

Trace: Torque T is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

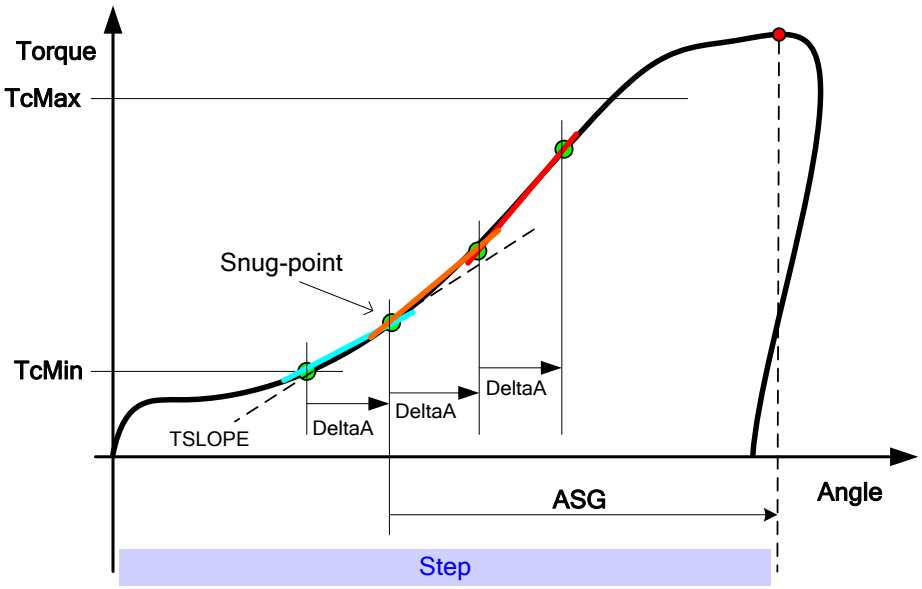
6.5.2.25 SG - Snug Gradient

Step type Id: 32

Parameters: DeltaA (Delta Angle), TcMIN (Min Torque), TcMAX (Max Torque), TSLOPE (Torque Slope), ASG (Angle run), DIR (Direction).

Function:
1. The spindle is started with the programmed speed in direction DIR.
2. The search for the Snug-point is started as soon as the torque passes TcMIN.
3. The average slope over DeltaA deg is calculated.
4. This is repeated for every DeltaA deg.
5. As soon as two slopes after each other are larger than TSLOPE the Snug-point is found.
6. From this point, the Snug-point, the spindle runs another ASG deg.
7. If the torque level TcMAX is passed without finding the Snug-point the step is aborted and an event is generated.

Diagram:



TcMax	Max Torque	**ASG**	Angle run
TcMin	Min Torque	**DeltaA**	Delta angle
TSLOPE	Torque slope		

Result var.: After the step has finished the angle from the Snug-point to the end of the step can be checked using the normal angle check (see Check angle) if using the special start condition "SGP" (Snug Point). If so, the result is reported as the step level result variable "A".

Trace: Angle when ASG has been run from the detected Snug-point is shown as a horizontal line from step start to step stop in trace when Angle vs. Time is selected. Control parameters must be enabled in the trace form (see View of trace curves).

6.5.2.26 D - Diagnostic Step

Step type Id: 22

Parameters:

Step type	D - Diagnostic
Syncronize	☑
Start/restart monitoring	☐
Speed (rpm)	30
Spindle Functional Test	☑
Zero offset	Spindle Only (Recommended)
Angle count test	☐

Function: Defines the diagnostic checks to perform in a tightening program.

Normally the diagnostic step is the first step in a program but it can be a later step in the program (e.g. when Flying Zero Offset is used).

The diagnostic step can also be performed in a special program, which can be run between normal cycle programs. The normal programs will hence be faster.

The limits for PowerMACS zero offset compensations are configured using the Spindle Set Up form.

Select if you want a **Spindle Functional Test** to be performed. This test verifies the spindle is working correctly and adjusts the spindle's measuring system. This test is performed automaticly at startup but it is recommended to run a Diagnostic step with the spindle functional test as the first step in every tightening program to compensate torque measurements.

The spindle takes care of zero offset compensation, if this is not sufficient you can select to let PowerMACS *also* compensate for zero offsets using the **Static**, **Dynamic** or **Flying** method.

Check **Angle count test** to execute a test of the angle measurement channels. Chapter: Dynamic Zero Offset and Angle Count describes how this is done.

Errors detected in a diagnostic step are reported as **Fatal** errors except for one case. If the Flying Zero Offset parameter **Continue to run after failed check** is checked (see Spindle Set Up), a flying zero offset error is reported as a warning instead of as an error.

Note! The double torque transducer test is not executed while the spindle functional test is performed or zero offset is being measured. The double angle encoder test is not executed while the spindle functional test is performed.

Trace: There are no control parameters for this step type in trace.

6.5.2.27 BCT – Backlash correction (only available for Gauging)

Step type Id: 36

Parameters: **T** (Torque), **TI** (Delay time between directional change, **DIR** (direction to run),

Function: This step measures the lash between the angle transducer on the spindle and the external load (a bolt or another object that shall be rotated). It does so by running in both directions until a specified torque is reached and measures the angle between the end points. The angle measurements in all following steps are compensated with the measured angle every time the spindle changes direction.

The measurement is done on the angle channel specified as angle channel for Backlash correction in Program Settings.

1. Run spindle, in opposite direction of **DIR** until torque is **T**.
2. Wait time **TI**.
3. Run spindle in direction **DIR** until torque is **T** and measure the angle.

Trace: There are no control parameters for this step type in trace.

Note: The position registers used by the step Run to Position will be reset by the BCT step.

Result var: This step produces the step level result variable “BCT A” which is the angle from the end of the wait time to torque TC. This angle is used for backlash transportation in following steps.

6.5.2.28 POS - Run to Position (only available for Gauging)

Step type Id: 37

Parameters: **Type** (1 or 2),
Reference position (Null, Half, Peak, Low)
Position correction (relative to selected position reference, positive or negative)
Auto slow down,
Slow down position (Angle to reduce speed at, if not Automatic slow down)
Low speed (Speed used for last part of movement)

Type 2 parameters:
Adjust to (360 or 720 deg), **Min Angle** (Minimum angle to run)

Function, type 1:
Run the spindle to an absolute position stored in an earlier step. The target position of the step is the selected **Reference position** adjusted with the specified **Position correction.** The Position correction can be both positive or negative, meaning after or before the stored reference position.

Depending on the current position and the calculated target the spindle may run forward or reverse to reache the target position.

Function, type 2:
Run spindle in direction of previous step. The step calculates the necessary movement to

transport to the closest multiple (determined by **Adjust to**) of the distance to target position. This position is then adjusted with the **Position correction** parameter to get the final target of the step.

If the total movement is less than **Min Angle** another 360 or 720 degrees are added to the target position.

Slow down at step end:

When the spindle has reached an angle of **Slow down position** before the target position, the speed is reduced to **Low speed,** using the speed ramp programmed as “Speed ramp up” on the “Ramps & Other” tab.

If **Automatic slow down** is used the slow down angle is automatically calculated so the speed reaches 1 rpm just before the target position.

Trace: There are no control parameters for this step type in trace.

Example, type 2:

Reference position = Null position
Adjust to = 720 deg
Position correction = -100 deg
Min Angle = 50 deg

Null position is at 324 deg (from cycle start)
Current angle is 843 deg, i.e. 519 deg from Null position
Adjust to = 720 deg, i.e. the target position is 324 + 720 = 1044 deg
Position correction = -100 deg, i.e. the compensated target is 1044 – 100 deg = 944 deg

The total movement in the step is thus 944 – 843 = 101 deg. If this had been less than Min Angle (50 deg) another 720 deg had been added to the target.

6.5.3 Step – Restriction

You can set up restrictions that decide when a step shall be interrupted. A restriction stops a running step immediately. A restriction can be considered repairable or fatal.

All restriction checks except for the Fail Safe Torque limit starts when torque spike elimination has ended and stops after the step overshoot time has ended.

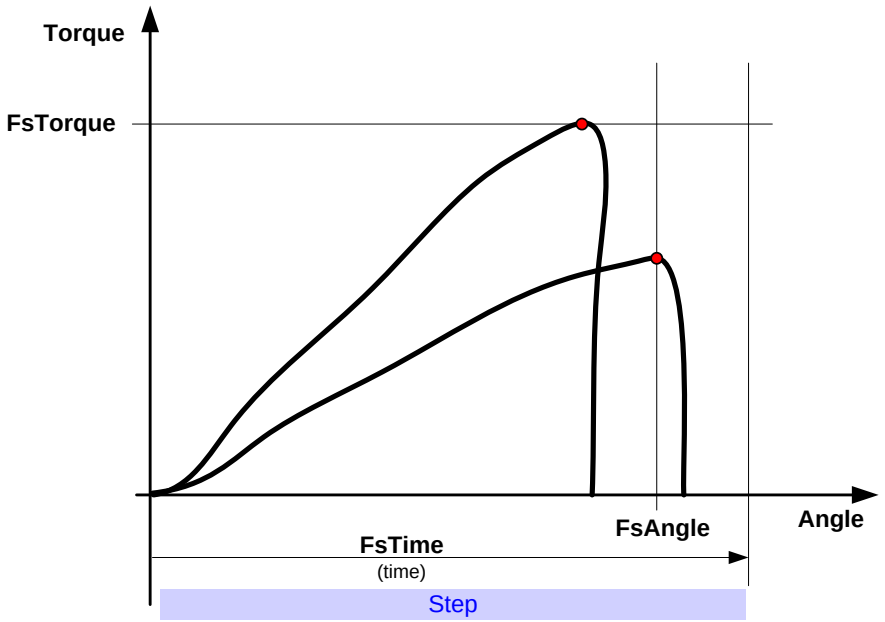
6.5.3.1 Fail Safe Torque

Parameters: Torque

Function: Supervises the fail-safe limits for Torque, The fail safe limits are active also during the step overshoot time.

The absolute value of the measured torque must be lower than the fail-safe torque limit during the entire step, including the post step check time. Note! The torque restriction is supervised also during the Torque Spike Elimination phase.

Diagram:



FsTorque	Fail safe torque	**Angle**	Fail safe angle
FsTime	Fail safe time		

Alarms: Torque restriction (TR)

Trace: Torque is shown as a horizontal line from step start to step stop in trace when Torque vs. Time is selected. Fail safe limits parameters must be enabled in the trace form (see View of trace curves).

6.5.3.2 Fail Safe Time

Parameters: Time

Function: Supervises the fail-safe limits for Time, The fail safe limits are active also during the step overshoot time.

Time measurement starts at step start (after start delay for step) and end when the step has ended and post step check time has elapsed. The measured time must be less than the fail-safe time limit.

Diagram: See Fail Safe Torque

Alarms: Time restriction (TIR)

Trace: Fail safe limits parameters must be enabled in the trace form (see View of trace curves).

6.5.3.3 Fail Safe Angle

Parameters: Angle

Function: Supervises the fail-safe limits for Angle, The fail safe limits are active also during the step overshoot time.

The angle is measured either from the current position where the step is started (AS1) or from the point where the configured Torque start level has been reached. The measured angle must be less than the fail-safe angle limit. No check is made until the selected start condition has been achieved.

Diagram: See Fail Safe Torque

Alarms: Angle restriction (AR)

Trace: Angle is shown as a horizontal line from step start to step stop in trace when Angle vs. Time is selected. Fail safe limits parameters must be enabled in the trace form (see View of trace curves).

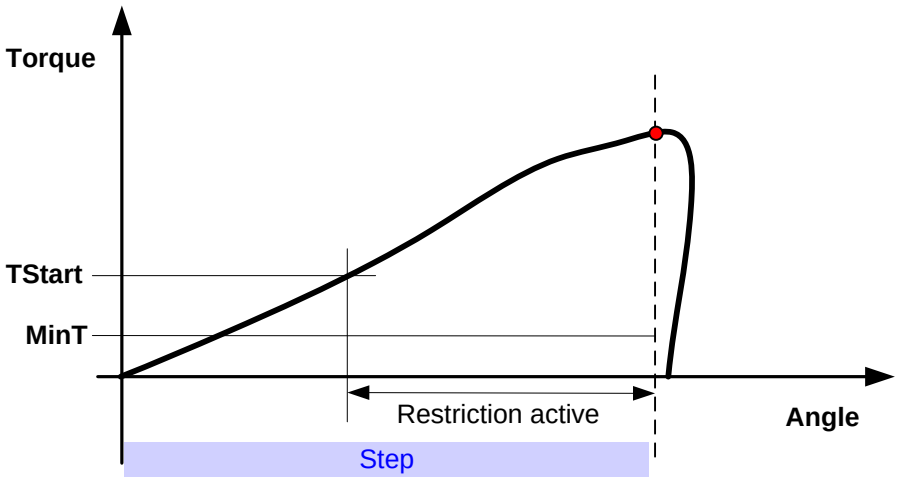
6.5.3.4 Min Torque Restriction

Parameters: Threshold torque TStart, Min torque limit MinT

Function: The restriction starts when the measured torque passes TStart. After this the step is stopped immediately if the measured torque falls below the level MinT. The restriction is only active until the control function of the step has shut off, since the torque normally drops when the spindle is stopped.

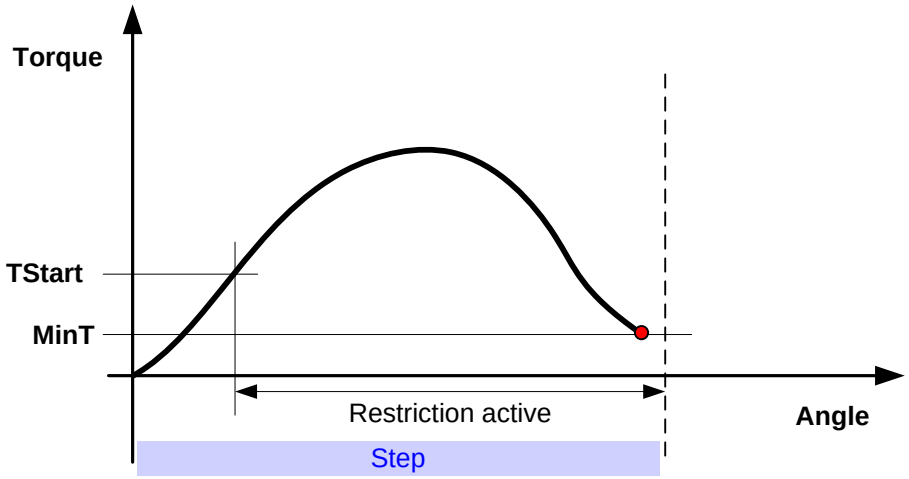
Note that the restriction is not active during the torque spike elimination phase in the beginning of the step. The search for TStart starts then the torque spike elimination is finished.

Diagram 1:



TStart	Threshold Torque
MinT	Min Torque Limit

Diagram 2:



TStart	Threshold Torque
MinT	Min Torque Limit

Alarms: Min Torque Restriction (MTR) if the torque drops below MinT

Trace: MinT is showed in trace as a horizontal line from TStart to step stop when Torque vs. Angle is selected. Restriction limits must be enabled in the trace form (see View of trace curves).

6.5.3.5 Gradient

Parameters: Threshold Torque, Angle distance, No degrees, Gradient Low, Gradient High

Function: Calculates and checks the gradient of the torque curve, that is, the ratio of torque vs. angle.

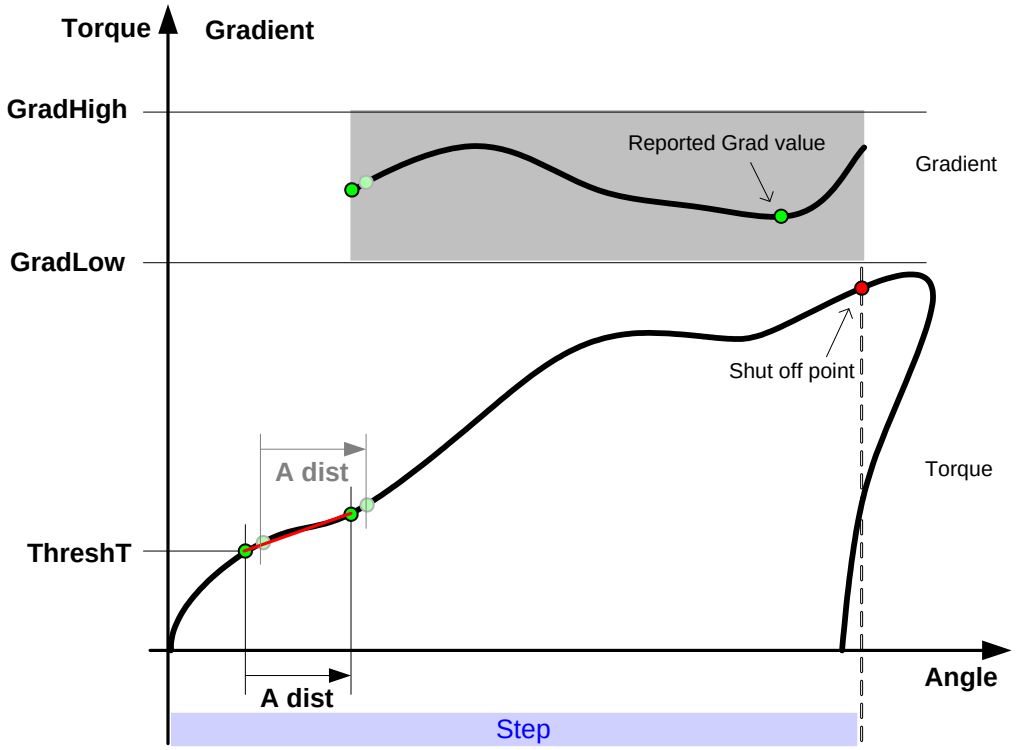
The gradient is calculated as the difference in torque Angle distance apart divided by the Angle distance. The first gradient is calculated Angle distance after Threshold torque has been reached. A new gradient is calculated every time a new torque and angle sample is available, that is, normally once every millisecond.

The calculated gradient is compared against the limits and stored in trace as a gradient curve. One or both of the gradient limits can be left empty. If both are empty no restriction check is active, but the calculated gradient is still stored in the trace.

The torque samples used for the gradient calculation are filtered using a sliding window average filter. The No degrees parameter defines the sliding window angle over which mean values are calculated.

In case the step runs backwards compared to the step direction the restriction ends the step if limit is specified. If no limit is configured the gradient calculation is restarted and there is a gap in the trace curve until the first new gradient can be calculated.

Diagram:



GradHigh	Gradient high	**ThreshT**	Threshold Torque
GradLow	Gradient low	**A dist**	Angle distance

Result var.: This restriction produces the step level result variable "Grad"which is the lowest gradient measured during the restriction or the gradient that violated the high limit of the restriction.

Alarms: Gradient high restriction (GHR) and Gradient low restriction (GLR).

Trace: Gradient High and Gradient Low are shown in trace as horizontal lines from the point where the first gradient was calculated to shut off when Gradient vs. Angle is selected. Restriction limits must be enabled in the trace form (see View of trace curves).

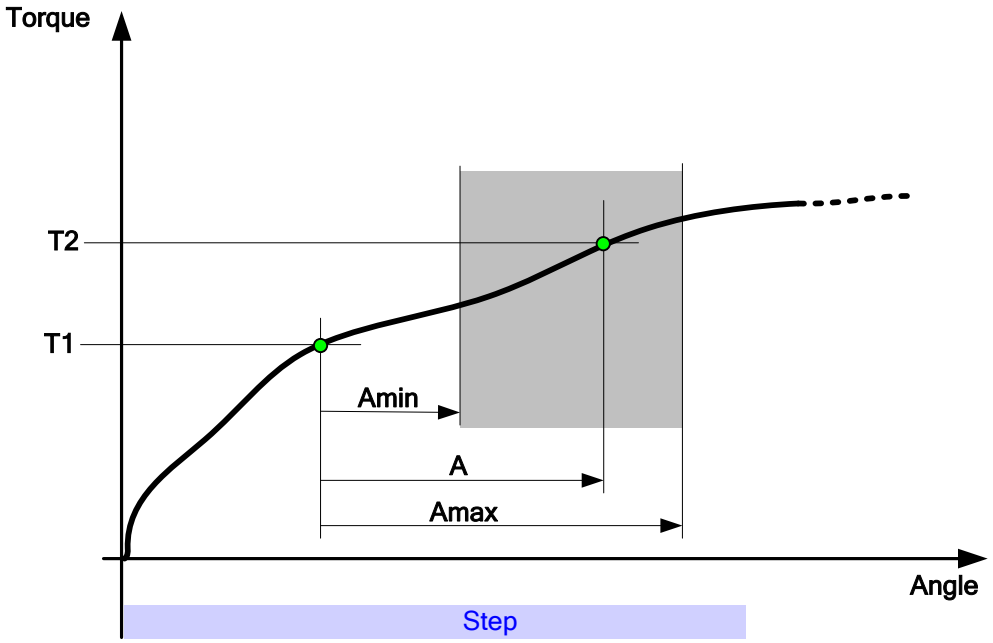
6.5.3.6 Cross Thread and Gradient

Parameters: Torque level T1, Torque level T2, Min angle, Max angle

Function: Measures the angle from the point where torque passes Torque level T1 to the point where torque passes Torque level T2. Checks that the measured angle, A, is between Min angle and Max angle.

Min angle or Max angle can be left empty meaning that the respective test is disabled. Note that at least one of them must be configured.

Diagram:



T1	Torque Level T1	**Amin**	Angle min
T2	Torque Level T2	**Amax**	Angle max
A	Measured angle from T1 to T2		

Alarms: Cross thread restriction (CROSSTR) if A > Max angle. Cross gradient too high restriction (CROSSGR) if A < Min angle.

Trace: Two lines are shown in Torque vs. Angle view. The horizontal line represents the T2 level and it starts when T1 is passed and ends after Amin degrees. The vertical line represents the point where Amax degrees are passed after T1. The step fails if any of the lines is crossed. Restriction limits must be enabled in the trace form (see View of trace curves).

6.5.3.7 Torque Profile

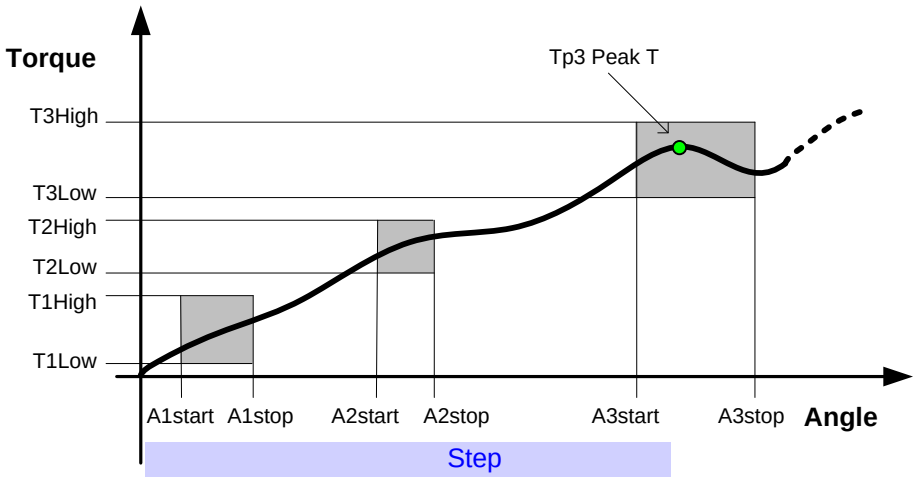
Parameters: TnStart, AnStart, AnStop, TnHigh, TnLow, where n is in the range [1..3].

Function: Checks that the torque in the three angle windows defined by TnStart and the pairs AnStart and AnStop, is within the unique limits TnHigh and TnLow. The angles are measured from from the start point according to the step angle definition if TnStart isn't configured and from TnStart if the parameter is used.

TnStart, AnStart and/or AnStop may be left empty. Leaving AnStart empty corresponds to the beginning of the step while an empty AnStop corresponds to the end of the step.

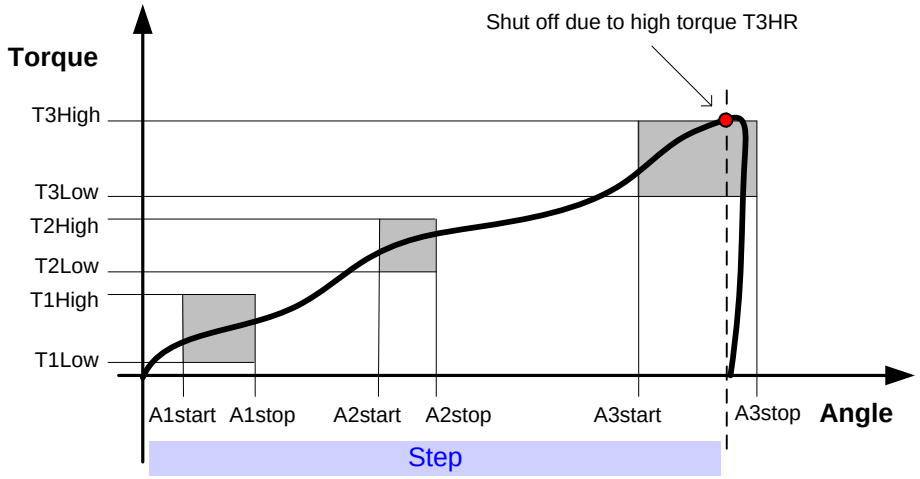
TnHigh and/or TnLow may be left empty. Leaving them empty disables the test against the limit in question. Note that if none of the torque limits is specified then TnStart, AnStart and AnStop must also be empty and the restriction is not used.

Diagram 1:



TnLow	Torque Low	**AnStart**	Angle start
TnHigh	Torque High	**AnStop**	Angle stop

Diagram 2:



TnLow	Torque Low	**AnStart**	Angle start
TnHigh	Torque High	**AnStop**	Angle stop

Result var.: This restriction produces the step level result variables "Tp1 Peak T", "Tp2 Peak T" and "Tp3 Peak T" which is the highest torque measured in the respective interval.

Alarms: Torque in window n too high (TnHR) and/or Torque in window n too low (TnLR).

Trace: TnHigh and TnLow are shown in trace as horizontal lines from AnStart to AnStop when Torque vs. Angle is selected. The step fails if any of the lines is crossed. Restriction limits must be enabled in the trace form (see View of trace curves).

6.5.3.8 Angle Difference

Parameters: Use spindle limits, Max difference, Number of samples

Function: Supervises Angle channel 1 readings compared to Angle channel 2 readings. The restriction requires two angle channels to be equipped on the spindle.

Select if you want to use the spindle's stored limits or configure your own limits (**Max difference** – Maximum difference in degrees, **Number of samples –** For how many samples (milliseconds) Max difference is allowed).

If a angle double transducer protection is configured on the spindle this restriction will replace that during the step, see Angle Double Transducer Check.

Diagram: None

Alarms: Double Angle Transducer Error (ADIFF)

Trace: -

6.5.3.9 Torque Difference

Parameters: Use spindle limits, Channels, Max difference, Number of samples

Function: Supervises two torque channels compared to each other.

Select if you want to use the spindle's stored limits or configure your own limits (**Max difference --** Maximum difference in Nm and **Number of samples –** For how many samples (milliseconds) Max difference is allowed).

If a torque double transducer protection is configured on the spindle this restriction will replace that during the step, see Torque Double Transducer Check.

Diagram: None

Alarms: Double Torque Transducer Error (TDIFF)

Trace: -

6.5.4 Step – Check

Checks are used to define which step level variables to measure and optionally test against programmable limits.

Note! If a check of a specific variable is not defined the variable will not be measured and therefore not available for reporting. This also goes for the corresponding SPC values.

The **Overshoot time** is configured for each step (see Ramps & Other – Other). This parameter decides for how long the measuring should continue after that the control function for the step has shut off.

All checks can be set to repairable (R, default) or fatal (F). If repairable the Reject Management can try to re-tighten the joint (see Step – Rejects). Select the **Fatal** box if you want the fault to be considered **Fatal**.

All found check errors in a step are reported when the step has finished. Checks do not interrupt a running step.

The limits of the most checks are optional. A check having no limit defined will always generate an OK result. This might be useful if its only purpose is to produce a result variable to be reported. Check limits are available for display in trace for many of the checks. Check limits must be enabled in the trace form (see View of trace curves).

Note! If the conditions required for evaluation of a check for some reason are not fulfilled an event saying so is generated. Should the check have one or more limits defined then the corresponding error bits are set as well and the status of the checks is set to **Fatal** independent of the configuration. An example of such conditions is that the start condition for a Check angle check is not met.

In the following sub chapters all checks are described in detail together with their parameters.

With the parameter **Operate on** (only avaliable in Gauging) it is possible to select if the check shall work on the Step or Zone level. It is a combo box that has the selections ‘Step’ and all available zones for the step. E.g. if you have specified 3 zones for the step the selections will be ‘Step’, ‘Zone 1’, ‘Zone 2’, ‘Zone 3’. Some of the checks can only work on the step level, see the table below.

With the parameter **Start condition** it is possible to select the start condition for the check. The available choices vary between the different check types and if the check operates on step or zone level, see the table below (if a check only have one possible start or stop condition the parameter will not be shown).

A check not running in a zone will start as soon as the backlash correction is finished (if the step run backlash correction, otherwise the check start directly at step start).

Check Type	Operate on	Start condition (operate on Step)	Start condition (operate on Zone)	Condition for OK (operate on Zone)
Check Peak T	Step or Zone	Step start Speed reached	Zone start	Zone started
Torque in Angle window	Step or Zone	Step start	Zone start	Zone started

Torque in Time window	Step or Zone	Step start	Zone start	Zone started
Check Mean T	Step or Zone	Angle Torque Time Step start	Angle Torque Time Zone start	Zone ended
Check Angle	Step or Zone	*Start condition:* Torque Step start (AS1) Shut off previous (AS2) Peak angle previous (AS3)	*Start condition:* Torque Zone start	Zone started
		Stop condition: Shut off Shut off previous Peak Angle Final Angle Torque	*Stop condition:* Zone end Torque	Zone ended (if stop condition is Zone End)
Check Post view Torque	Step	Step start		
Check T/T3	Step or Zone	Step start	Zone start	Zone ended
Check shut off Torque	Step or Zone	Step start	Zone start	Zone ended
Check torque rate	Step or Zone	Step start	Zone start	Zone started
Check Time	Step or Zone	Step start	Zone start	Zone started
Check Yield point	Step	Step start		
Check Stick Slip	Step	Step start		
Check Delta T	Step or Zone	Step start	Zone start	Zone started
Low Spot T	Step or Zone	Step start Angle	Zone start Angle	Zone started
Check Position	Step	Step start		

6.5.4.1 Result values from checks

For each check it is possible to select the cycle data variable it shall use to store the result. Each check type has its own specialized variables that can only be used for that check type (e.g. **Peak T, Peak T1, Peak T2** and **Peak T3** for a peak torque check). There are also general variables available that can be used for all check types.

The specialized variables can only be used in the correct check type with the correct channel selection for torque or angle. For example **Peak T** can only be used in a peak torque check with the torque channel set to **Auto**. **A2** can only be used by an angle check with the angle channel set to **Angle 2**, etc.

Two sets of general variables are available, one on step level and one on bolt level, with 80 variables in each set. The variables on step level are named **Step var. 01**, **Step var. 02**, etc, and the variables on bolt

level are named **Bolt var. 01**, **Bolt var. 02**, etc. The first 40 variables of each type are marked with a **T**, to indicate that these are scaled according to the torque unit selection in the reporters.

It is also possible to select **Not reported**, which mean that no value will be reported for the check.

Checks that run in a zone, i.e. have **Operate on** set to **Zone**, can not use the **Specialized variable** for reporting, only the **Step var** and **Bolt var** are avaliable.

Note: It is only possible to use the same variable once in each step. For example, with two Peak Torque checks on the same step with channel selection Torque 1, only one can use the specialized variable Peak T1, the other one must use one of the general variables (or not reported). Setup problems will detect any errors and generate warnings if needed.

6.5.4.2 Check peak torque

Parameters: **TH** (Torque high), **TL** (Torque low),
Set null position (Store peak position to Null position register) (only available for Gauging)
Set peak position (Store peak position to Peak position register) (only available for Gauging)

Function: The peak torque from check start to check stop is checked to be between the high limit **TH** and the low limit **TL**.

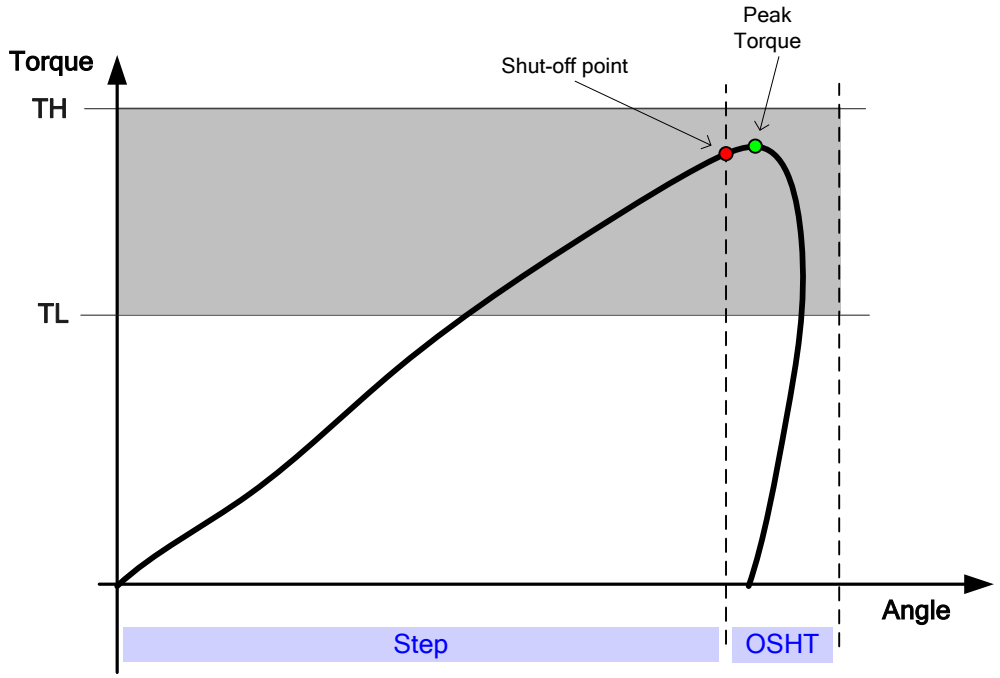
Both limits are optional. Leaving them blank disables the check but the result value is still calculated and reported.

If the check operates on step it starts at step start. It ends when the step ends, including overshoot time. If the check operates on zone the check starts when the selected zone starts and ends when the zone ends.

If **Set null position** or **Set peak position** is set the peak position, that is the angle where the peak torque was found, is stored in the corresponding position register. It is possible to return to this position later with the step **POS - Run** to Position (only available for Gauging).

This check requires that at least one sample is collected after the torque spike elimination phase. If not, and at least one limit is specified, the steps status is set to **Fatal** and the error bit of the respective limit is set.

Diagram:



TH	Torque High	**OSHT**	Overshoot time
TL	Torque Low		

Result var.: Peak T, Peak T1, Peak T2, Peak T3 (Peak torque).

Alarms: Step peak torque high (PTH) and step peak torque low (PTL).

Trace: TH and TL are shown in trace as horizontal lines from step start to step stop when post step check time has elapsed. The lines are only shown when Torque vs. Time is selected.

6.5.4.3 Check torque in angle window

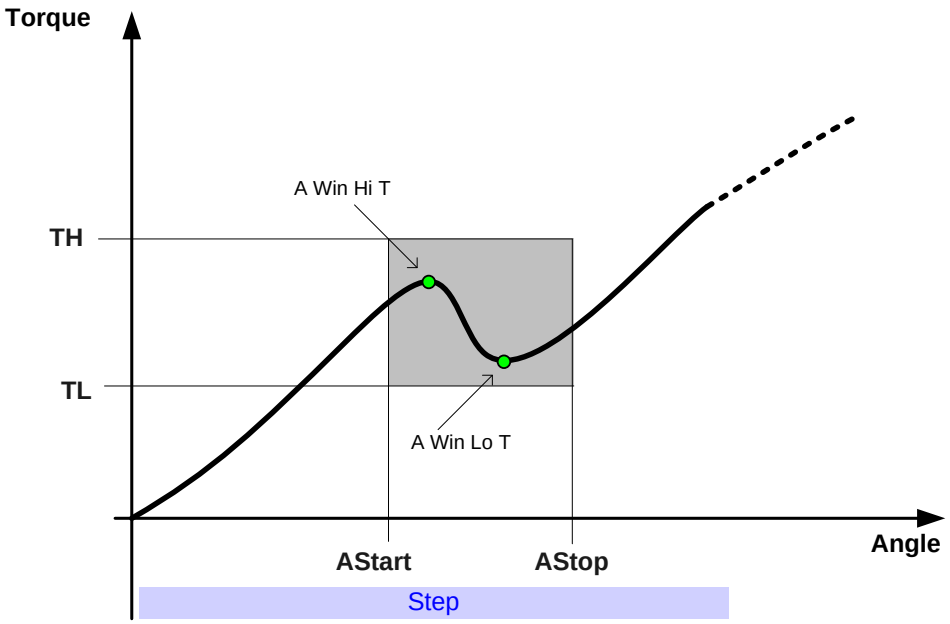
Parameters: **TH** (Torque high), **TL** (Torque low), **Astart**, **Astop**

Function: **Astart** and **Astop** are both measured from the start of the step and define an angle window. All torque values measured in this window are checked to be between the high limit **TH** and the low limit **TL**.

Both limits are optional. Leaving them blank disables the check but the result values are still calculated and reported.

This check requires that at least one sample is collected after the torque spike elimination phase and that angle **Astop** is reached. If not, and at least one limit is specified, the steps status is set to **Fatal** and the error bit of respective limit is set.

Diagram:



TH	Torque High	**AStart**	Angle start
TL	Torque Low	**AStop**	Angel stop

Result var.: "A Win Hi T" (angle window high torque) and "A Win Lo T" (angle window low torque).

Alarms: Step torque high in angle window (THAWIN) and step torque low in angle window (TLAWIN).

Trace: TH and TL are shown in trace as horizontal lines from AStart to AStop. The lines are only shown when Torque vs. Angle is selected and Check limits are enabled in the trace form (see View of trace curves).

6.5.4.4 Check torque in time window

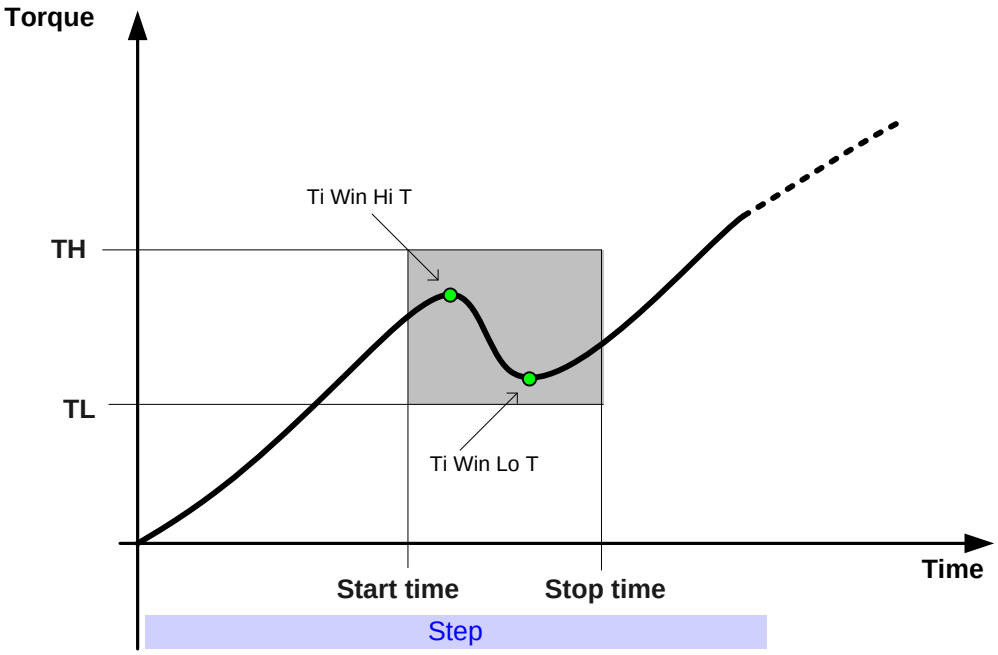
Parameters: **TH** (Torque high), **TL** (Torque low), **Start time**, **Stop time**

Function: Start time and Stop time are both measured from the start of the step and define a time window. All torque values within this window are checked to be between the high limit TH and the low limit TL.

Both limits are optional. Leaving them blank disables the check but the result value is still calculated and reported.

This check requires that at least one sample is collected after the torque spike elimination phase and that time Stop time is reached. If not, and at least one limit is specified, the steps status is set to **Fatal** and the error bit of the respective limit is set.

Diagram:



TH	Torque High	**Start time**
TL	Torque Low	**Stop time**

Result var.: "Ti Win Hi T" (time window high torque) and "Ti Win Lo T" (time window low torque).

Alarms: Step torque high in time window (THTIWIN) and step torque low in time window (TLTIWIN).

Trace: TH and TL are shown in trace as horizontal lines from Start time to Stop time. The lines are only shown when Torque vs. Time is selected and Check limits are enabled in the trace form (see View of trace curves).

6.5.4.5 Check mean torque

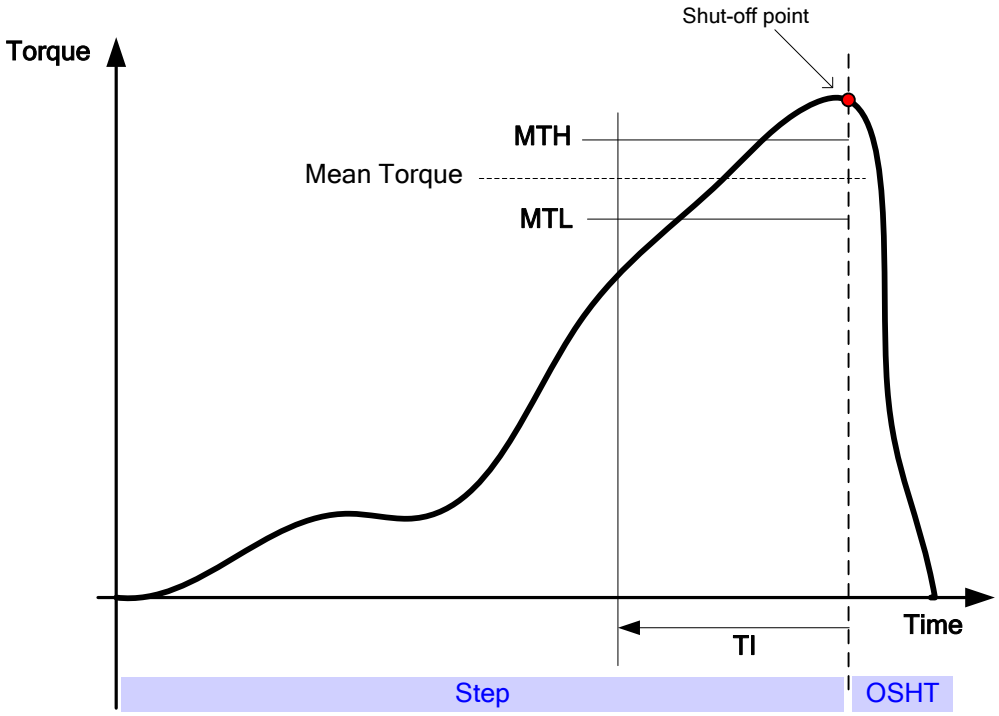
Parameters: **MTH** (Mean torque high), **MTL** (Mean torque Low), **TI** (Time)

Function: The average torque during the last TI seconds of the step is measured. The calculated value is checked to be between the high limit MTH and the low limit MTL.

Both limits are optional. Leaving them blank disables the check but the result value is still calculated and reported.

This check requires that at least one sample is collected after the torque spike elimination phase. If not, and at least one limit is specified, the steps status is set to **Fatal** and the error bit of the respective limit is set.

Diagram:



MTH	Mean Torque High	**OSHT**	Overshoot time
MTL	Mean Torque Low		
TI	Time		

Result var.: "Mean T" (mean torque).

Alarms: Step mean torque high (MTH) and Step mean torque low (MTL).

Trace: There are no limits in trace for this check.

6.5.4.6 Check angle

Parameters: **AH** (Angle high), **AL** (Angle low), **Start condition**, **Tstart** (Torque start), **Stop condition**, **Tstop** (Torque stop), **Operate on** (Step or Zone),
Torque channel (Auto, Torque 1, Torque 2 or Torque 3 (Current as Torque)),
Angle channel (Auto, Angle 1 or Angle 2), Result variable

Function: Measures the angle movement made during a step and checks that it is between the high limit **AH** and the low limit **AL**. When to start and stop the measuring is controlled using parameters **Start condition** and **Stop condition** as follows:

Start condition	Value used as start angle
T	The angle, A_Tstart, measured when torque reaches the value **Tstart**.
AS1	The angle of the current position when the step is started.
AS2	The angle measured at shut of the previous step.
AS3	The peak angle reached during the execution of the previous step.
SGP	The "Snug Point "of a SG step. See SG - Snug Gradient for a description.
Zone start	Angle at zone start

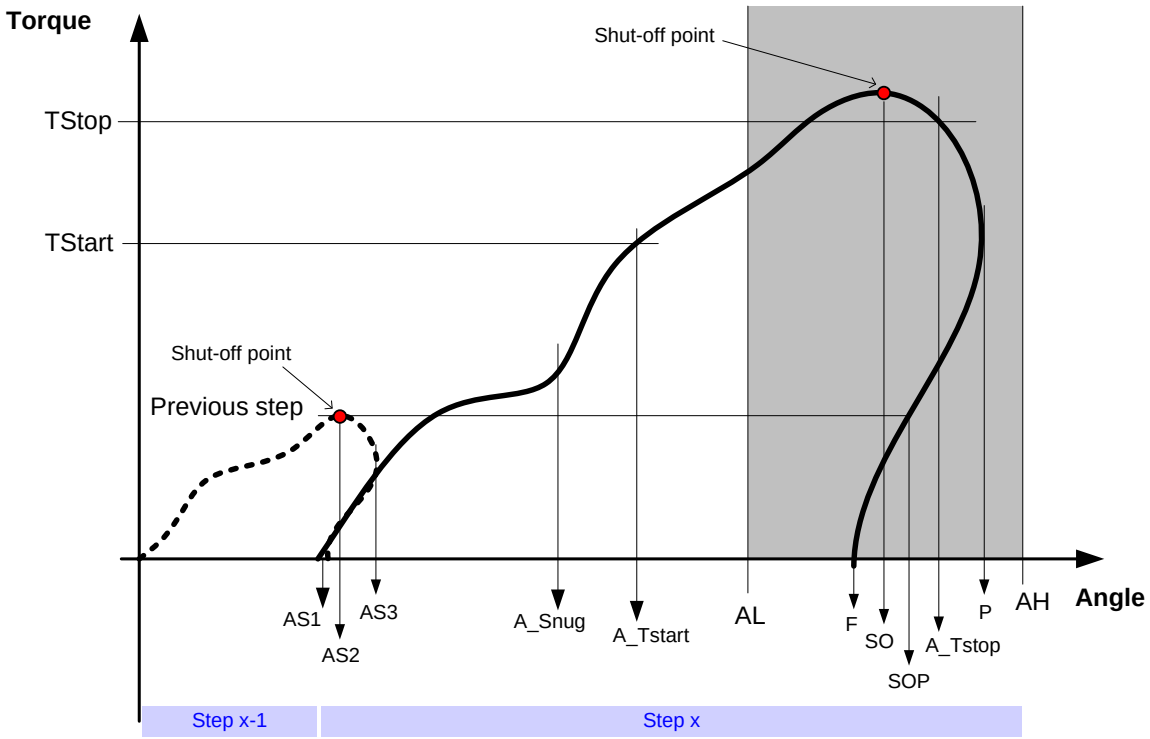
Stop condition	Value used as end angle
F - "Final"	The final angle (AE1, measured when the "Overshoot time" has passed after shut off).
SO - "Shut Off"	The angle measured at the shut of point (AE2).
P - "Peak"	The largest angle measured from shut off until the "Overshoot time" has passed (AE3).
SOP - "Shut Off Previous"	The angle measured when the torque falls under the shut off torque measured during the previous step (AE4).
T - "Torque"	The angle, A_Tstop, measured when torque drops below the value of **Tstop** after the shut off point.
Zone end	Angle at zone end

Note! The "Overshoot time" (which is measured from that the step is shut off, see chapter: Step – Check) has to be programmed long enough to monitor the requested end condition.

Both limits are optional. Leaving them blank disables the check but the result value is still calculated and reported.

The check is only evaluated if both the **Start condition** and the **Stop condition** are met. If not, and at least one limit is specified, the steps status is set to **Fatal** and the error bit of the respective limit is set.

Diagram:



TStart	Torque Start	**A_Tstart**	Start cond. - Angle at Tstart	**F**	Stop cond. - Final angle
TStop	Torque Stop	**AS1**	Start cond. - Angle at step start	**SO**	Stop cond. - Shut-off angle
AL	Angle Low	**AS2**	Start cond. - Angle at prev. shut off	**P**	Stop cond. - Peak angle
AH	Angle High	**AS3**	Start cond. - Peak angle prev.	**SOP**	Stop cond. - Shut-off angle prev.
		A_Snug	Start cond. - Snug point	**A_Tstop**	Stop cond. - Angle at Tstop

Result var.: A (step angle)

Alarms: Step angle high (AH) and Step angle low (AL).

Trace: AH and AL are shown in trace as horizontal lines from check start to stop, which vary according to selected start and stop conditions. The lines are only shown when Angle vs. Time is selected and Check limits are enabled in the trace form (see View of trace curves).

6.5.4.7 Check Post view Torque

Parameters: **TC** (Torque check level),
TL (Torque low), **AdistL** (Low angle distance), **AwinL** (Low angle window),
NOSL (No samples low)
TH (Torque high), **AdistH** (High angle distance), **AwinH** (High angle window),
NOSH (No samples high)

Function: This function checks that the filtered torque in the angle window AwinL is higher than the torque limit TL. It also checks that the filtered torque in the angle window AwinH is less than the torque limit TH. Both limits are optional but at least one of them must be specified.

Angle window AwinL is located AdistL before the point where the torque reaches level TC and AwinH is located AdistH before the torque reaches level TC. The torque level is allowed to reach level TC several times during a step. The angle windows are measured from the last point where the torque reached TC.

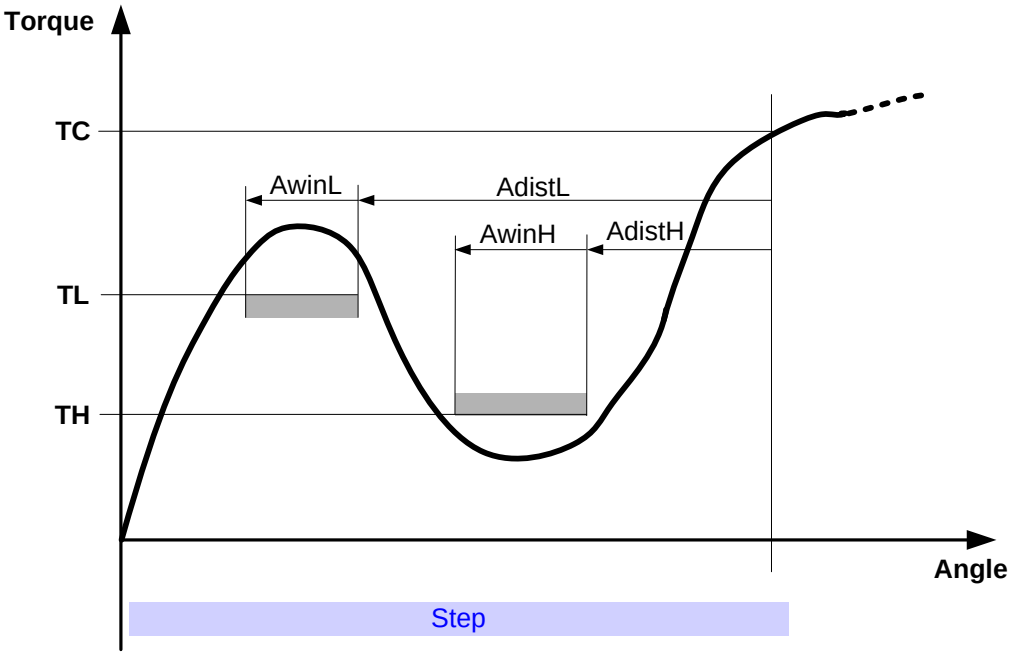
The torque values used in the check are filtered using a sliding window average algorithm. The mean torque value is calculated over NOSL for the TL check and over NOSH for the TH check. The filter can be disabled by leaving the respective number of samples parameter empty.

The filter starts after the torque spike elimination phase and runs throughout the check. The check does not start until the filter is full. The filtered torque value is recalculated on every new sample; that is, normally every millisecond.

The angle used for crossings over the limits is the filter's median angle value (angle at sample NOSL/2 or NOSH/2 in the filter).

This check requires that torque TC is reached after the torque spike elimination phase and that the angle MAX(AwinL + AdistL; AwinH + AdistH) has been reached at that point. If not, the steps status is set to **Fatal** and the error bit of the respective limit is set.

Diagram:



AwinL	Low Angle Window	**TL**	Torque Low
AwinH	High Angle Window	**TH**	Torque High
AdistL	Low Angle Distance		
AdistH	High Angle Distance	**TC**	Torque Check Level

Result var.: None.

Alarms: Post view torque too high (PVTH) and post view torque too low (PVTL).

Trace: TH and TL are shown in trace as horizontal lines displaying the angle window for each limit. The lines are only shown when Torque vs. Angle is selected and Check limits are enabled in the trace form (see View of trace curves).

6.5.4.8 Check T/T3 (Current as Torque)

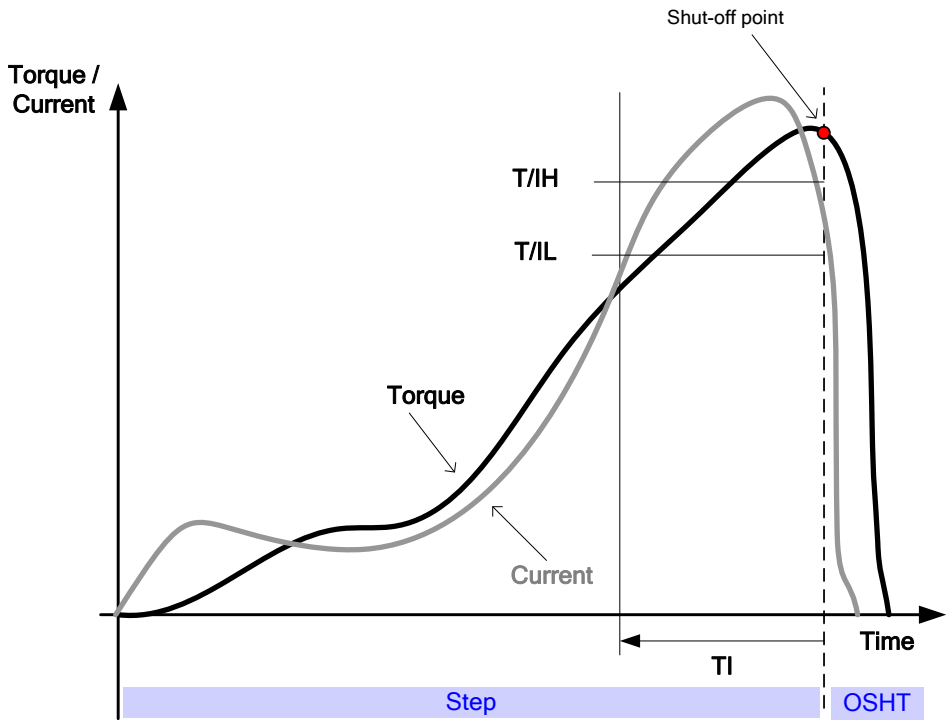
Parameters: **T/IH** (Torque/T3 high), **T/IL** (Torque/T3 low), **TI** (Time)

Function: The average torque vs. the average current measured during the last TI seconds of the step is measured and checked to be between the high limit T/IH and the low limit T/IL.

Both limits are optional. Leaving them blank disables the check but the result value is still calculated and reported.

This check requires that at least one sample is collected after the torque spike elimination phase. If not, and at least one limit is specified, the steps status is set to **Fatal** and the error bit of the respective limit is set.

Diagram:



T/IL	Torque/Current Low	**OSHT**	Overshoot time
T/IH	Torque/Current High		
TI	Time		

Result var.: "T/T3"

Alarms: “T/T3 high” and “T/T3 low”.

Trace: There are no limits in trace for this check.

6.5.4.9 Check shut off torque

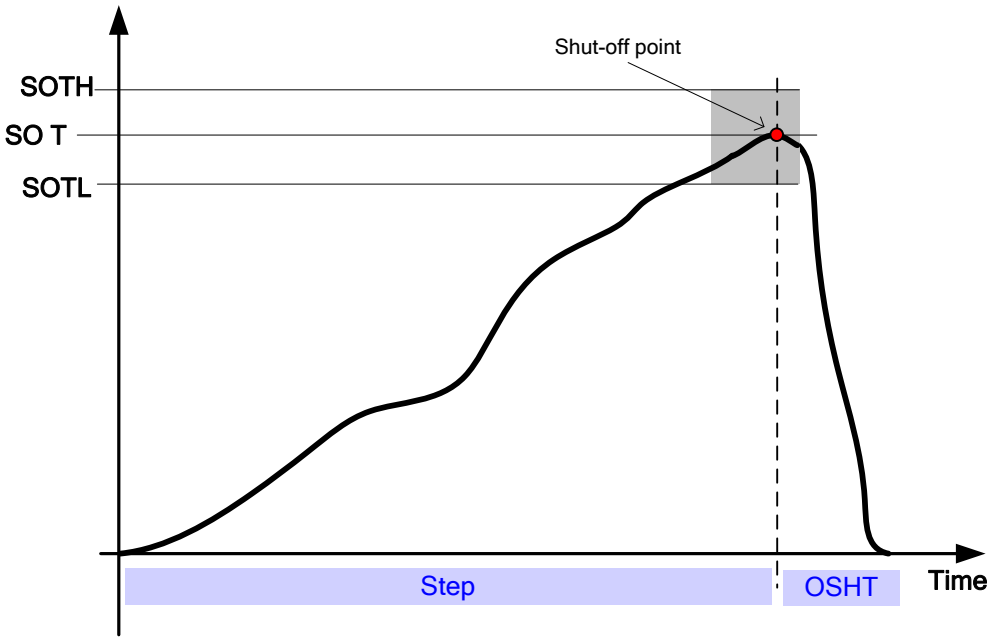
Parameters: **SOTH** (Shut off torque high), **SOTL** (Shut off torque low)

Function: The torque measured at the shut off point of the step is checked to be between the high limit SOTH and the low limit SOTL.

Both limits are optional. Leaving them blank disables the check but the result value is still calculated and reported.

This check requires that at least one sample is collected after the torque spike elimination phase. If not, and at least one limit is specified, the steps status is set to **Fatal** and the error bit of the respective limit is set.

Diagram:



SOTH	Shut-Off Torque High	**SO T**	Shut-Off Torque
SOTL	Shut-Off Torque Low	**OSHT**	Overshoot time

Result var.: "ShutOff T" (shut off torque).

Alarms: Step shut off torque high (SOTH) and step shut off torque low (SOTL).

Trace: TH and TL are shown in trace as horizontal lines from step start to step stop after post step check time. The lines are only shown when Torque vs. Time is selected and Check limits are enabled in the trace form (see View of trace curves).

6.5.4.10 Check torque rate

Parameters: **T Start** (Torque Start), **T Stop** (Torque Stop), **A Start**, **A Stop**, **TRL** (Torque rate low), **TRH** (Torque rate high) , **Dev** (Deviation)

Function: Measures and checks the Torque Rate (TR), that is, the ratio of torque vs. angle during a single step. The check is performed the same way as the Monitoring function

Monitoring Check – with the following differences:

The check uses it's own buffer for recording torque versus angle. This buffer has 4096 elements (instead of 8192) which each represent a given angle interval in units of [encoder increments/buffer position]. The buffer is auto scaled, meaning that it starts with resolution 1 [encoder increment/buffer position] and whenever the buffer limit is reached this value is doubled giving lower resolution but longer measuring angle.

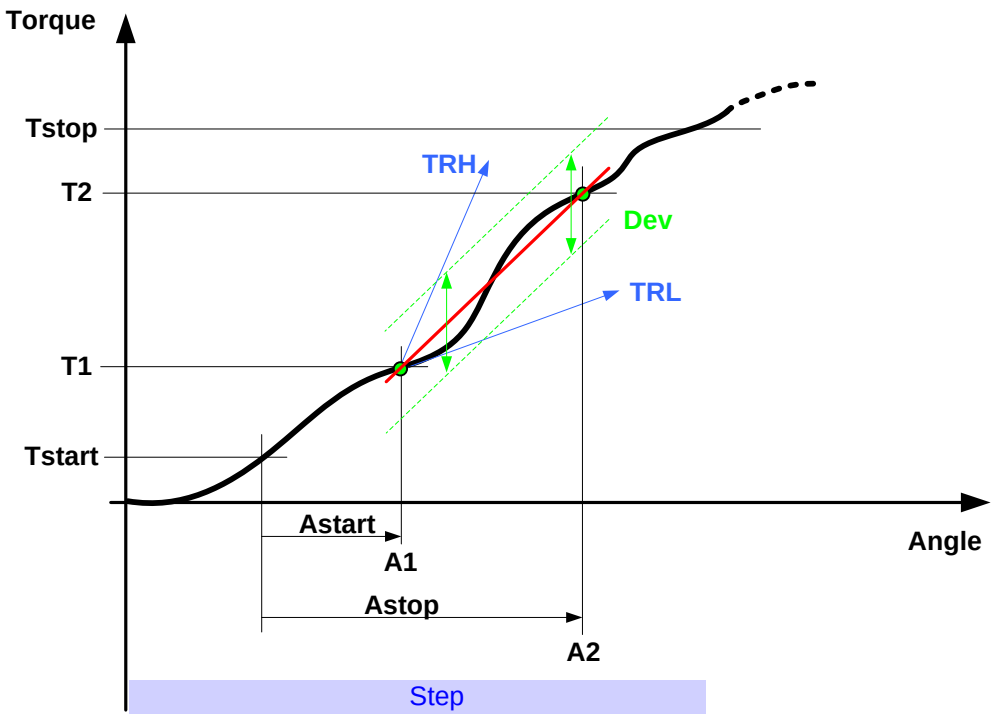
The buffer is **not** filtered for maximum torque for increasing angle before the checks are evaluated.

The start condition is looked for from the point where the torque spike elimination phase is over. If T Start is empty measuring is started when torque becomes larger than zero. This means that negative torque rates can be measured and that the Torque Rate limits can be negative.

Only one interval can be specified.

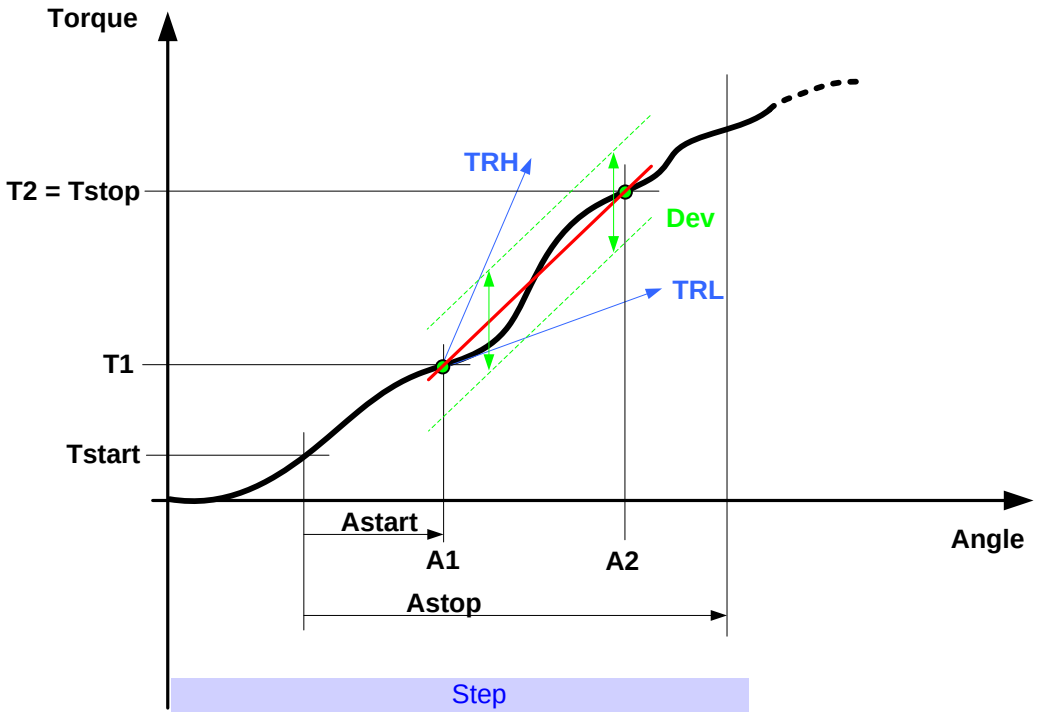
This check requires that the start condition and, if specified, the stop condition are met. Also, the angle measured between these two points must be larger than the angle corresponding to one (1) encoder increment. If not, and at least one limit is specified, the steps status is set to **Fatal** and the error bit of the respective limit is set.

Diagram 1:



Tstart	Torque Start	**Astart**	Angle Start	**Dev**	Deviation
Tstop	Torque Stop	**Astop**	Angel Stop	**TRL**	Torque Rate Low
				TRH	Torque Rate High
T1	Torque at the end of Astart				
T2	Torque at Astop				

Diagram 2:



Tstart	Torque Start	**Astart**	Angle Start	**Dev**	Deviation
Tstop	Torque Stop	**Astop**	Angel Stop	**TRL**	Torque Rate Low
				TRH	Torque Rate High
T1	Torque at the end of Astart				
A2	Angle at Tstop				

Result var.: "TR" (step torque rate) and "TR Dev" (step torque rate deviation).

Alarms: Torque rate too low (TRL), Torque rate too high (TRH) and Too big deviation (DEV)

Trace: TRH and TRL are shown in trace as lines originating from T1 showing the selected torque rate. The TRH and TRL lines end where either T2 or Astop should have been reached using the specified torque rates (see above diagram). The Dev lines are shown as in the diagram above. The lines are only shown when Torque vs. Angle is selected and Check limits are enabled in the trace form (see View of trace curves).

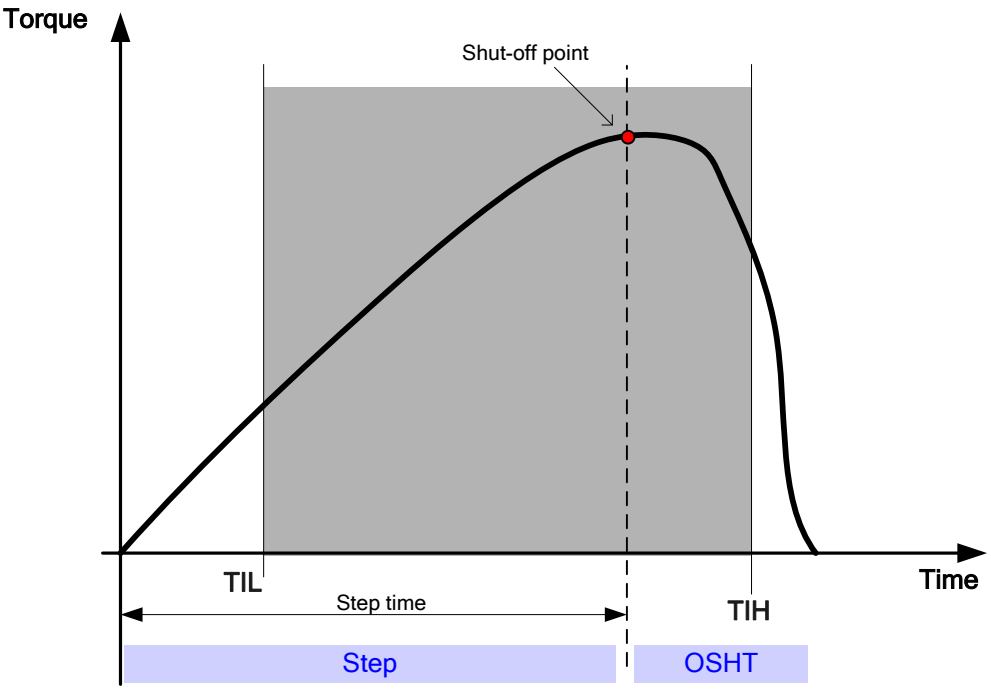
6.5.4.11 Check time

Parameters: **TIH** (Time high), **TIL** (Time low)

Function: The time from when the step starts until it ends is measured and checked to be between the high limit TIH and the low limit TIL.

Both limits are optional. Leaving them blank disables the check but the result value is still calculated and reported.

Diagram:



TIL	Time Low	**OSHT**	Overshoot time
TIH	Time High		

Result var.: "Time" (step time).

Alarms: Step time high (TIH) and Step time low (TIL).

Trace: There are no limits in trace for this check.

6.5.4.12 Check Yield Point Angle

Parameters: Yield Angle high, Yield Angle low, Torque start, No degrees, Angle distance, DIFF, Stop condition

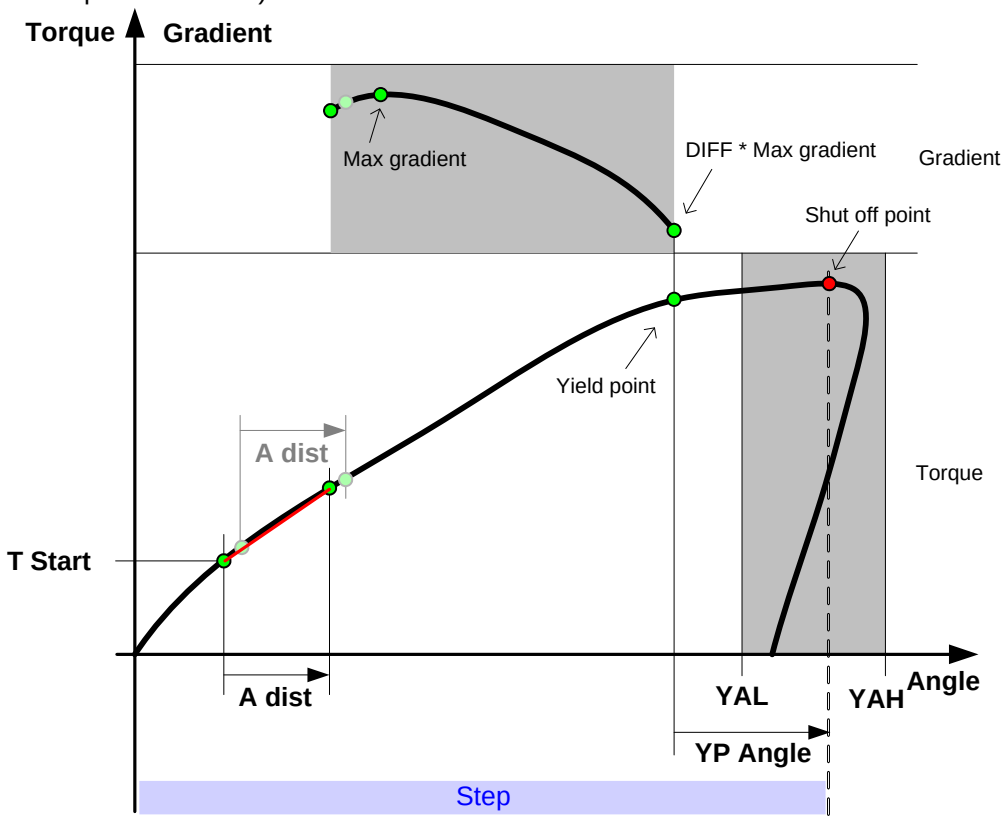
Function: This check measures the angle from the yield point to the end of the step. The measured angle, YP Angle, is checked to be within the limits Yield Angle low and Yield Angle high. If the value is outside an error is reported.

The search for the yield point starts then the torque has reached the level Torque start. From this point the gradient, i.e. the ratio of torque vs. angle, is calculated. During the step the maximum gradient value is stored. The yield point is found then the calculated gradient is less than DIFF * Maximum gradient. The DIFF value is specified as a percentage of the maximum gradient.

The angle to use as end angle is controlled by the parameter Stop condition. There are three different angles that can be used. Final angle, F, is the last measured angle in the step. Shut off angle, SO, is the angle at the shut off point and Peak angle, P, is the highest angle reached during the step. For more details see Check Angle.

The gradient calculation is done in the same way as in the Gradient Restriction. The check compares the filtered torque values Angle distance apart to find the gradient. The torque is filtered with a sliding average over the angle No degrees. See also the Gradient Restriction for more details.

Diagram: (with Stop condition SO)



T Start	Torque start	**A dist**	Angle distance
YAH	Yield point angle high	**DIFF**	Difference %
YAL	Yield point angle low		

Result var.: "YP Angle" (Yield Point Angle), the angle between the Yield point and the end angle.

Alarms: Yield Point Angle High (YAH) and Yield Point Angle Low (YAL)

Trace: No limits are shown in trace.

6.5.4.13 Check Stick Slip

Parameters: Trigger mode, Max drops, Trigger level, Threshold torque.

Function: This check detects and reports stick slip effects during a step. The detection is done by counting the times the torque falls below the level TTrig and if the number of times is larger than Max drops the stick slip error is reported.

The check operates in two different modes, Dynamic or Fix. The difference between the modes is the way TTrig is calculated.

Dynamic mode: TTrig is dynamically calculated as percentage of the current max torque that has been reached so far during the step. The user specifies the percentage to use. The check is started as soon as the torque passes the threshold torque level.

Fixed Mode: TTrig is a fixed torque level specified by the user. In this mode the Threshold torque is not needed.

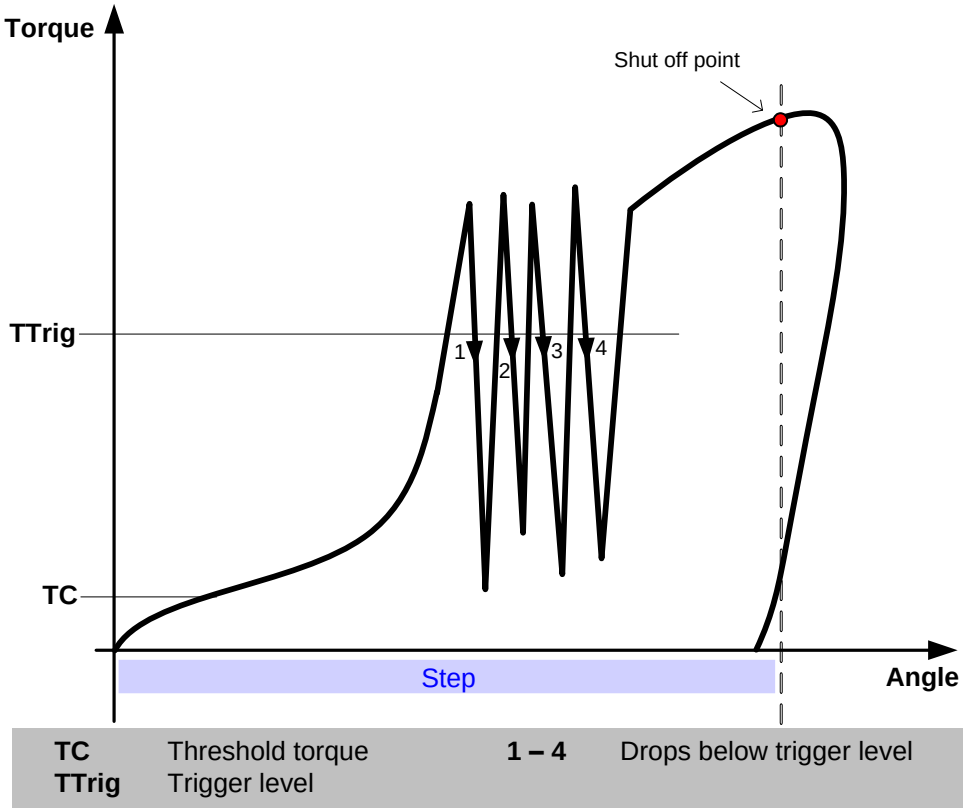
To prevent false detections of stick slip the counter of the drops below TTrig is reset if a too long angle has been run since the last drop. The counter is reset if the angle since the latest drop is larger than the angle between the first and latest drop, i.e. if

$a - a_i > a_i - a_1$; a = current angle, a_1 = angle at first drop, a_i = angle at last drop

Note 1: Since this is a check the step will continue to run to the final target even after a stick-slip has been detected.

Note 2: Since the sample frequency of the torque transducer normally is 1 kHz the check can have problems to detect a stick slip with a frequency higher than 500 Hz.

Diagram:



Result var.: none

Alarms: Stick slip detected (SS)

Trace: The torque trigger level is shown as a horizontal line from the first drop to the last. In dynamic mode the line represents the maximum trigger level that was used. The line is only shown when Torque vs. Angle is selected.

6.5.4.14 Delta T (only available for Gauging)

Parameters: **TH** (Maximum delta torque), **TL** (Minimum delta torque).

Function: The difference between the peak torque and the low spot torque, measured from check start to check end is compared to the limits. **TH** is the high limit and **TL** is the low limit. Both or only one of the limits can be programmed.

If the check operates on step it starts at step start. It ends when the step end, the last sample is taken at the shut off point of the step. If the check operates on zone the check start when the selected zone starts and ends when the zone ends.

Result var: Delta T

Alarms: Delta torque high (DTH) and Delta torque low (DTL).

Trace: No limits are shown in trace.

6.5.4.15 Low Spot Torque (only available for Gauging)

Parameters: **TH** (Maximum low spot torque), **TL** (Minimum low spot torque),
Start condition, **Astart** (for Start condition Angle)
Set null position (Store low position to Null position register)
Set low position (Store low position to Low position register)

Function: The lowest torque during the check is measured and compared to the limits (TH, TL). The measurement starts when the angle **Astart** (measured from check start) has been passed. If **Start condition** is **Step start** or **Zone start** the measurement starts directly when the check starts.

The measurement ends then the step or zone ends. If it is running on step level the last sample is taken at the shut off point of the step.

If **Set null position** or **Set low position** is set the low position, that is the angle where the low spot torque was found, is stored in the corresponding position register. It is possible to return to this position later with the step **Run to Position**.

Result var: Low Spot T

Alarms: Low spot torque high (LTH) and Low spot torque low (LTL).

Trace: Torque vs. Angle

6.5.4.16 Check Position (only available for Gauging)

Parameters: **PH** (Maximum distance to selected position),
PL (Minimum distance to selected position),
Reference position (Null, Half, Peak or Low position),

Function: The distance to the selected position at the shut off point of the step is compared to the limits **PH** and **PL**. Both limits are optional.

The distance is calculated as Distance = Reference Position – Current Position, the result can be both negative and positive.

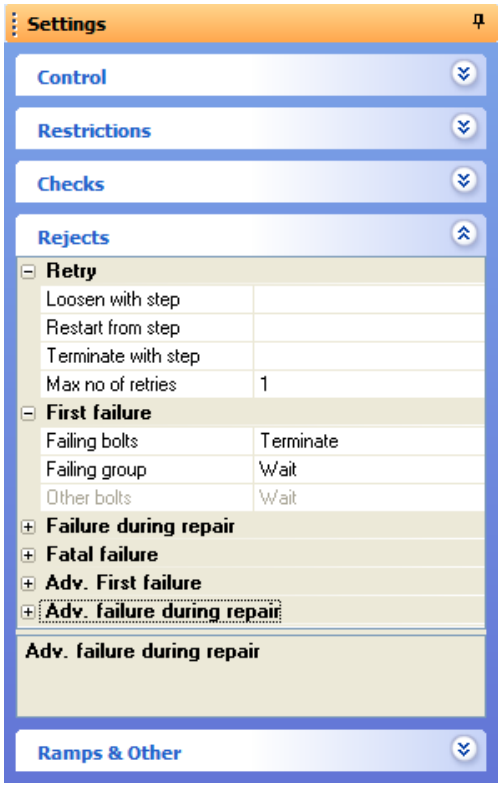
Result var: Position A

Alarms: Position high (PH) and Position low (PL).

Trace: No limits are shown in trace.

6.5.5 Step – Rejects

The *Reject Management* (RM) function is used for automatic repair of failed steps. Most of the RM settings for a step can be found in the **Rejects** group.



When a failed step is detected the RM performs the following:

1. Checks bolt status
2. Decides what to do
3. Starts repair
4. If problem was correctable, continues with status OKR
5. If not correctable, terminates or continues with status NOK

6.5.5.1 Checking the bolt status

The first thing that the Reject Management function does when a step fails is to figure out whether or not the step can be repaired. This is decided by the bolts current statuses.

At each synchronized step all bolts will be assigned a status. This status is the result of all checks made in the step, comprising the restriction tests, the step checks, hardware checks, etc. The resulting overall status for a bolt will be one of the following:

Status of all restrictions, checks, etc. of the step	**Bolt status**
All checks are Acceptable	Acceptable
At least one check is Repairable and none is Fatal	Repairable
At least one check is Fatal	Fatal

6.5.5.2 Deciding what to do

Whenever a reject management action is considered the following concepts are important:

Bolt Categories

All bolts executing are divided into one of three groups

- **Reject bolts**: All Bolts that have other status than Acceptable, that is have failed them self (Repairable or Fatal).
- **Reject group**: All Bolts that have status Acceptable but belong to a group in which at least one bolt is not Acceptable.
- **Others**: All Bolts that have status Acceptable and do not belong to a group where one or more bolts are not Acceptable.

Possible RM Actions

There are four different **RM Actions** to choose between:

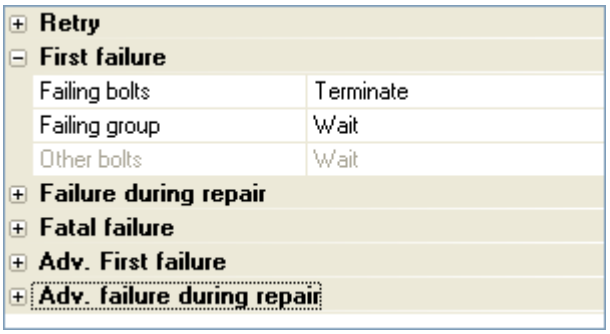
- **Terminate**: The bolts should execute their Terminate sequence.
- **Continue**: The bolts should continue with the next step. The bolt will first wait for any Terminate and Repair sequences to finish.
- **Retry**: The bolts should execute their Retry action(s). The bolt will first wait for any Terminate sequences to finish.
- **Wait**: The bolts should wait at the step they currently are at while other bolts are being terminated or repaired.

Selecting RM Action

You can specify different action for each bolt category depending on what kind of error that is at hand, and when it occurred. This is configured using the **First Failure**, **Failure during repair** and **Fatal failure** settings specified for the steps.

- The **First failure** alternatives are used when the station is in its Running Normal state and for cycle RM. It offers the following alternatives and combinations:

Failing bolts	Failing group	Other bolts
Terminate	Terminate	Terminate
Terminate	Terminate	Wait
Terminate	Wait	Wait
Continue	Continue	Continue
Retry	Retry	Retry
Retry	Retry	Wait
Retry	Wait	Wait

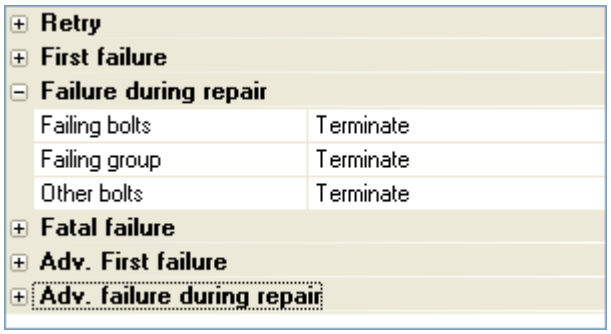


- The **Failure during repair** alternatives are used when the station is in its Running Repair state, that is when an error occurs during the repair.

  Please note that the Retry alternative is **not** available for the steps located in the repair part of the program. The steps in the repair part of the program are only intended for preparation of a repair. CE steps do not have **Failure during repair** since all CE steps finishes the cycle (and thereby ends any on-going repair).

  The Failure during repair action only affects bolts that are active running repair. Bolts that are waiting, or have finished their repair, are not affected. The following Failure during repair alternatives are available in the main program:

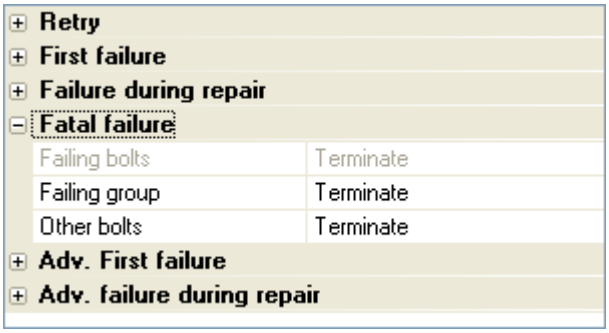
Failing bolts	Failing group	Other bolts
Terminate	Terminate	Terminate
Terminate	Terminate	Wait
Terminate	Wait	Wait
Continue	Continue	Continue
Retry	Retry	Retry



  Note! No Retry is allowed as **Failure during repair** action when steps in the repair part are run.

- The **Fatal failure** alternatives are used when a fatal error occurs for the bolt. It offers the following alternatives:

Failing bolts	Failing group	Other bolts
Terminate	Terminate	Terminate
Terminate	Terminate	Wait
Terminate	Wait	Wait



The Fatal failure action affects all bolts that are not already terminated. That is, also the bolts that are waiting while other bolts are being repaired may be ordered to Terminate.

Depending on the overall status of all bolts that have executed the station will choose Reject Management actions as follows:

Status of all Bolts	Reject Management Action
All bolts are Acceptable	Continue with next step
At least one bolt is Repairable and none is Fatal	Depends on the state of the station: • If it is Running Normal (that is not currently repairing) then the **First failure** settings of all bolts in the **Failing bolts** category are evaluated in order to decide the next action. • If it is already Running Repair then the **Failure during repair** settings of all bolts in the **Failing bolts** category are evaluated in order to decide the next action to take for the bolts running repair.
At least one bolt is Fatal	First the actions according to the **Fatal failure** settings of all bolts with status Fatal are executed. Note that also bolts that are waiting for other bolts to be repaired can be ordered to terminate using the Fatal try settings. If there are any bolts remaining with Repairable errors after running the terminate sequence they will be executed according to the **First failure** or **Failure during repair** settings as above.

Since the First failure, Failure during repair and Fatal failure settings are defined per bolt the station can meet contradictory actions. Should this happen the ambiguity is resolved by choosing the action in the following order: Terminate, Retry, Wait, and Continue.

When the station has decided what action each bolt should take it will order the bolts to either run the Terminate sequence, the Retry sequence, to continue with next step, or do nothing (wait).

The Retry and Terminate sequences are defined individually for each step and bolt. This means that if a bolt is ordered to execute a Retry action while at step 5, it is the Retry settings (see Configuring Reject Management) on step 5 that controls at which step the repair will start.

Note 1 If no Repair action is configured for a reject bolt the bolt will get status NOK instead of OKR after the repair. All other bolts will get status OKR as best after repair, independent of if any repair action was done or not.

Note 2 When a bolt is ordered to terminate the following rules are used to decide its status:

- A bolt that has a termination sequence defined, that is, the parameter **Terminate with step** is non-empty, for the last step it executed (in its current state) will get status NOKRM (NOK if it belongs to the Failing group).
- A bolt that ends its cycle before reaching a CE step due to a termination order will get status NOKRM (NOK if it belongs to the Reject bolts group).
- Remaining bolts, that is, those that have reached a CE step and does not have a termination sequence defined for the last step executed, will keep the status they have before they are ordered to terminate since the termination order will not cause any action.
- Emergency stopped bolts that do not have a termination sequence defined get status OK if all steps were run successfully and NOK if not.
- Emergency stopped bolts that have a termination sequence defined get status TERMNOK since the termination steps were not run.
- All bolts that fail during the termination sequence will get status TERMNOK.

Note 3 If a bolt is ready with initial repair, and has no repair step, it waits for the other bolts to finish their repair. If any of these other bolts fail when running the “Failing synchronization interval” the failing bolt will cause a repairable error treated as a **First failure** as the repair is ready. The waiting bolt will do Repair or Terminate according to the Retry in the step from which the repair was started.

Note 4 Bolts that end their cycle at a CE step in the repair part of the program reenters state Running Normal when the CE step is reached. The repair is finished when all bolts running repair have reached CE or the Failing synchronization interval. At this point the bolts go back to state Running Normal. If any bolt still running steps fails while a bolt waits after last step before CE, the waiting bolt will do Repair or Terminate according to the Retry in the step it is waiting at.

Please note that the above rules controls the status a bolt will have after executing the normal steps but before evaluating the monitoring checks configured for the CE step. See chapter: Cycle RM for a description of how monitoring affects the resulting status.

See also chapter: Reject management states for a description of the states of the station and its bolts with respect to reject management.

Synchronization marks

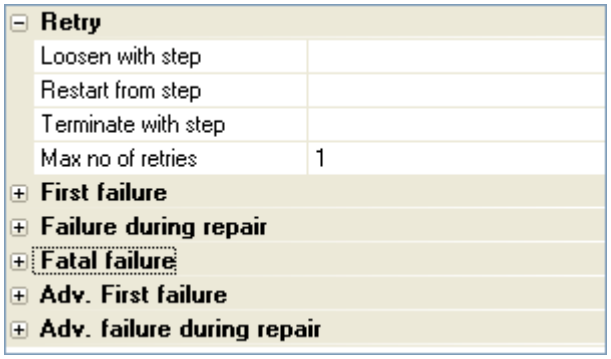
The usage of synchronization marks is very important for the execution of reject management. The station will only make decision on reject management action when all bolts have stopped either due to that they have failed executing a step or that they have reached a synchronization mark. They are also important with respect to when a bolt is considered done with any repair (see chapter: Reject management states).

In addition to the explicit synchronization marks there are also a number implicit ones, namely:

- All **Cycle End** (CE) and **Return from Repair** (RE) steps.

6.5.5.3 Configuring Reject Management

Most reject management settings are defined per step using the **Rejects** group.



The parameters grouped under **Retry** controls how this bolt should be repaired on the selected step when ordered so by the station:

- **Loosen with step**. The number of the step to jump to in order to prepare the repair action. Any step in the repair part of the program (that is, any of the steps after the first CE step) is allowed as target. Typically this sequence is used to loosen the bolt before the repair is started.

  If initiate repair steps end with a Cycle End step the cycle ends and the retry is considered ready. If left blank then no initial repair is done. That is, if a Cycle End step is reached while executing in the repair part of a program the cycle is ended for the bolt. Monitoring and cycle reject management is done according to the Cycle End step in the program.
- **Restart from step**. Points to the step in the main part of the program or sequence where the repair should be started when the optional initial repair is finished. Any step in the main part of the program up to, and including, the current step is allowed. If left blank execution resumes at the next step after the step where the Retry was ordered and repair is ready.
- **Terminate with step**. The number of the step to jump to when ordered by the station to Terminate. Any step in the repair part of the program is allowed as target.

  If the sequence started with the **Terminate with step** end with a CE step then monitoring will be performed as configured in this Cycle End step. If a RE step ends the sequence then monitoring is performed according to the CE step that should have been reached if the execution had not been transferred to the Terminate sequence.

  Whether to use a RE or a CE step should be based on if you want to use different Monitoring check settings for the Terminate sequence compared to the main sequence.
- **Max no. of retries**. Defines how many times this step may be repaired when it has failed. When this number is exceeded the error is considered **Fatal**. Max number of retries is 9.

If both **Loosen with step** and **Restart from step** are specified then **Loosen with step** sequence is executed first. If none of them are specified when ordered to do a repair the **reject bolt** will get status NOK instead of OKR as no repair has been done. All other bolts that are ordered to do a repair will get status OKR as best.

6.5.5.4 Advanced RM Actions

The RM actions specified above do not take into consideration any ”history” of the cycle. Decisions are made for each sequence separately, without looking back on what has happened earlier in the cycle.

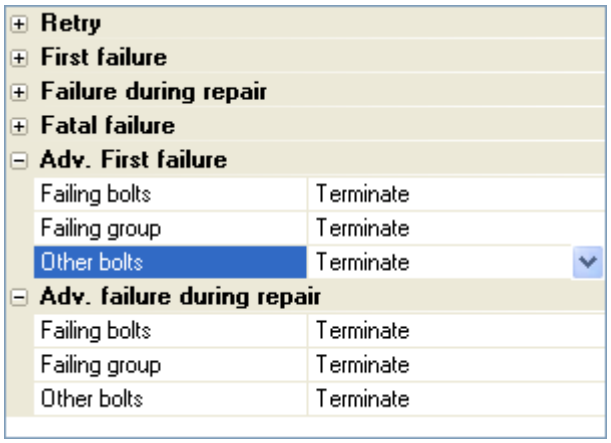
With the Advanced RM you can have an alternative RM dependent on what has happened earlier, in terms of number of sequences, spindles, and groups that have failed.

The conditions for when to start Advanced RM are set up for a Station using the System Map, see chapter Station Set Up.

You may choose one or more of the following alternatives (with step is here meant all steps between two synchronization points):

- When more than **No. of reject steps in cycle** erroneous steps have been detected during the cycle. The function counts the number of times steps fail (regardless of which bolt(s) that failed and which step number that failed) while the station is running in state "Normal". That is, whenever the First failure alternative is at hand.
- When more than **No. of reject groups in step** erroneous groups have been detected in the current synchronization interval. This function counts the number of groups that includes one or more failing bolts. The check is done after each synchronization point and involves only the bolts that executed the step(s). This means that bolts that where ordered to "Wait", "Continue" or "Terminate" in a previous step (or synchronization point) are not counted.
- When more than **No. of reject bolts in step** erroneous bolts have been detected in the current synchronization interval. This function counts the number of bolts that failed. The check is done after each synchronization point and involves only the bolts that executed the step(s). This means that bolts that where ordered to "Wait", "Continue" or "Terminate" in a previous step (or synchronization point) are not counted.

The Advanced RM will be used if **any** of the selected conditions has been met. The conditions are common to all steps in the cycle, independent of which is the current step.



6.5.5.5 Cycle RM

To enable Reject Management also on Monitoring errors all Monitoring checks can be configured as either Fatal or Repairable.

When all bolts have reported Cycle End to the station and nothing is to be repaired the station orders the bolts to do monitoring of the cycle. Then the bolt starts evaluation of the Monitoring conditions. The spindles will report the result of the Monitoring checks to the station and if at least one not already terminated spindle reports Repairable or Fatal reject management will be executed. The station uses the status and RM settings of all not terminated rejected bolts to determine the reject action. The station change state to Evaluating when monitoring starts, but re-enters state Running if Reject Management is started.

Note 1 For configuration of Reject Action on a CE step only the **First failure** and **Fatal failure** settings are available.

Note 2 No terminated spindles are involved in reject action for the cycle. This includes OK spindles that are terminated after the last step before CE without termination step and thus keeps status OK. Monitoring check results always affect the status of the individual bolt. OK or OKR bolts become NOK if monitoring fails and no reject management is done.

Bolts that have been terminated before they reach the CE step will **not** participate in Reject Management executed on the CE step. Bolts that have status NOK before the Monitoring checks are evaluated are considered Repairable. There are only two possibilities for a bolt to reach the CE step with status NOK. The first is that it as a reject bolt has been ordered to Continue (failing step was not repaired). The second is that it as reject bolt has been ordered to Retry, but had no retry configured (failing step was not repaired).

The status reported to the station after monitoring is constructed using the bolt status after executing the ordinary steps and the result from evaluating monitoring. A cycle that is NOK before monitoring cannot become OK due to correct monitoring checks.

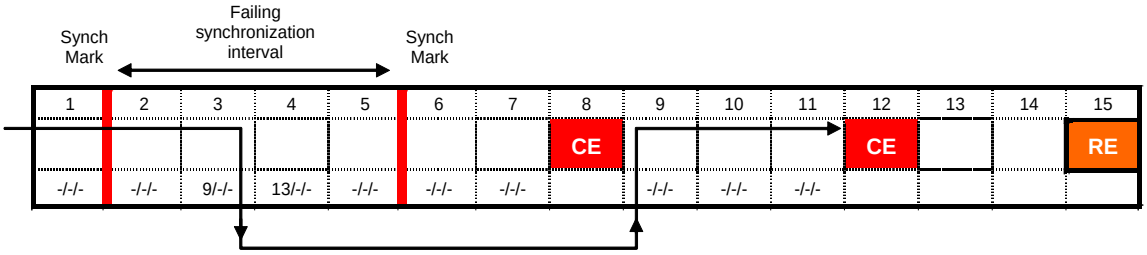
However, a bolt that has status NOKRM before monitoring is evaluated will keep the NOKRM status also if one or more monitoring checks fail.

The below table shows the possible outcome for the bolts involved in RM on the CE step (that is, all not already terminated bolts):

Status before evaluation of Monitoring checks	**Result of Monitoring checks**	**Reject Management action taken**	**Best possible status after repair**
OK	Acceptable	None	OK
	Repairable	According to First failure settings	OKR
	Fatal	According to Fatal failure settings	NOK
OKR	Acceptable	None	OKR
	Repairable	According to First failure settings	OKR
	Fatal	According to Fatal failure settings	NOK

NOKRM	Acceptable	According to First failure settings	OKR
	Repairable	According to First failure settings	OKR
	Fatal	According to Fatal failure settings	NOKRM
NOK	Acceptable	According to First failure settings	OKR
	Repairable	According to First failure settings	OKR
	Fatal	According to Fatal failure settings	NOK

Bolts that never reach a CE step, for example due to that it is terminated, will use the Monitoring checks defined on the CE step it would have reached if the execution should have continued normally. As an example of this see the program outlined below. If the bolt is ordered to terminate in step 2 the Monitoring settings of the first CE step (step 8) will be used. If terminated at step 10 the settings of the second CE step (step 12) is used.

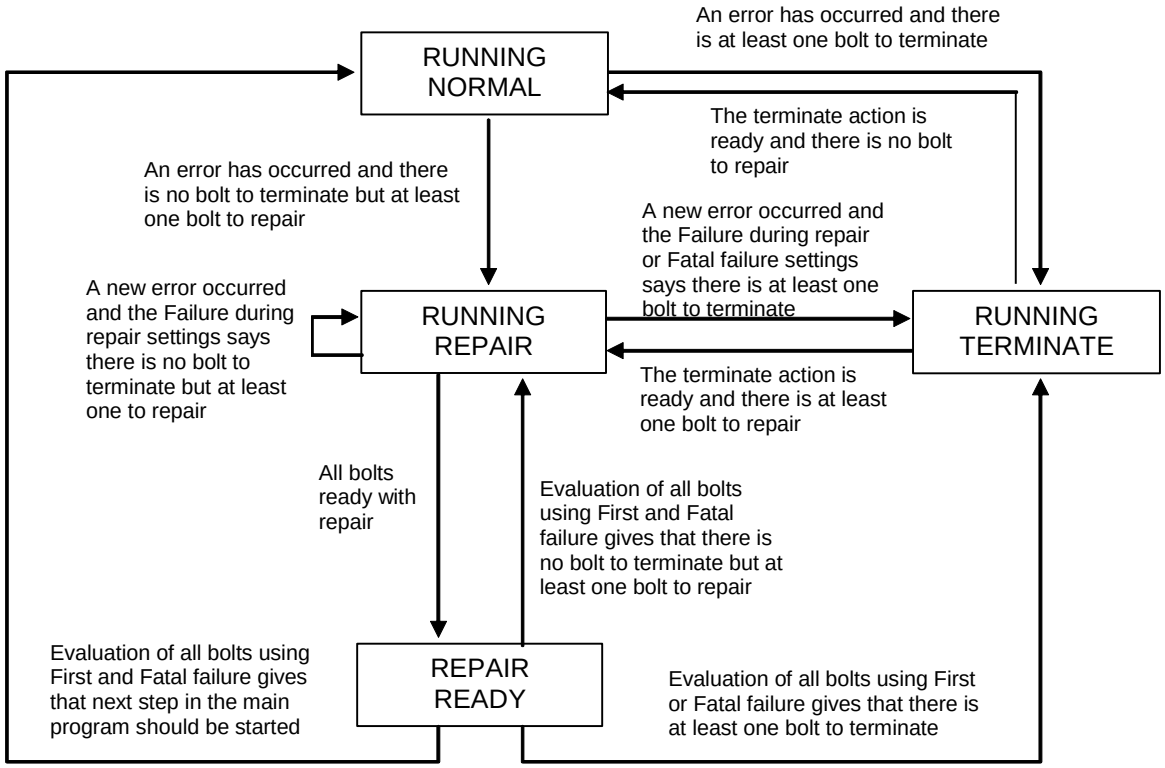


6.5.5.6 Reject management states

The states of the Station

With respect to reject management the station is always in one of the following states:

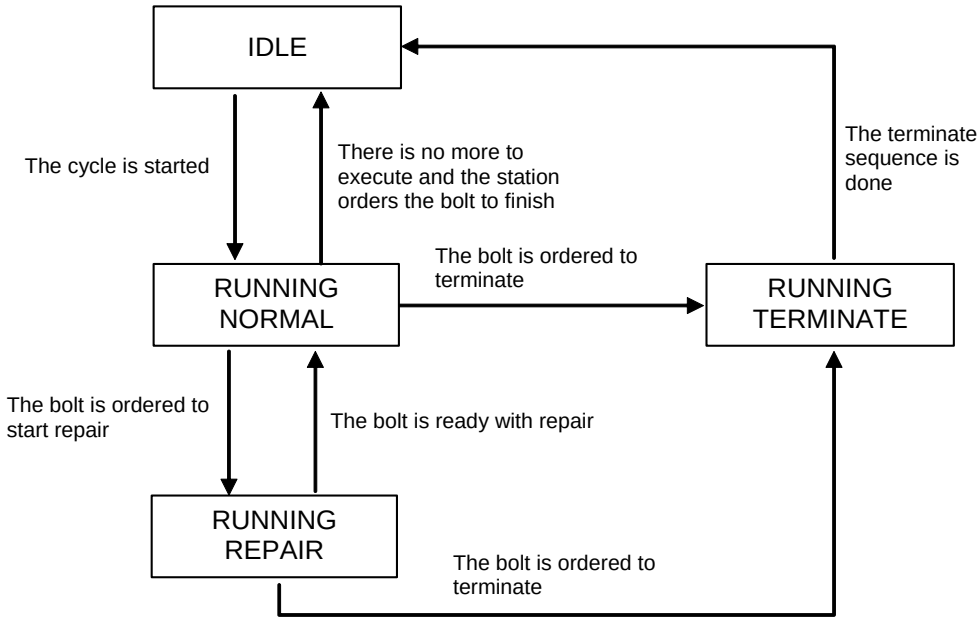
State	Description
Running Normal	All bolts are running steps in the main part of their programs. No error has occurred. When all bolts have stopped, either because they reached a synchronization mark or that they failed, the station decides what reject action to take by evaluating the • **Fatal failure** settings for all bolts with Fatal errors • **First failure** settings for all bolts with Repairable errors Should the settings lead to different actions for a given bolt the “hardest” action is taken
Running Repair	Some bolts are running their repair sequence. When all bolts have stopped, either because they reached a synchronization mark or that they failed, the station decides what reject action to take by evaluating the following conditions for all bolts that are not yet ready with their repair: • **Fatal failure** settings for all bolts with Fatal errors • **Failure during repair** settings for all bolts with Repairable errors **Note 1:** Only the bolts that currently are running repair are affected by any **Failure during repair** actions. Bolts that never entered the repair sequence, for example due to that they had Wait or Continue as reject action, and bolts that are already ready with the repair sequence are not involved. **Note 2:** The **Fatal failure** action affects all bolts, that is, also the bolts not currently Running Repair. Should the settings lead to different actions for a given bolt the “hardest” action is taken
Repair Ready	All bolts that once started their repair sequence have finished it. In this state the station do not execute any steps; it only evaluates the outcome of the repair in order to decide the next action. The station evaluates the reject settings of all bolts that started repair and was not terminated while executing it using the • **Fatal failure** settings for all bolts with Fatal errors • **First failure** settings for all bolts with Repairable errors The action decided is taken for all bolts that are not terminated, this means that not only the bolts that have done repair but also all bolts that had Wait or Continue as reject action when the station entered the Running Repair state may have new actions ordered here.
Running Terminate	One or more bolts are running their terminate action. Synchronization is maintained for these bolts as long as the steps end OK. Should a step fail then the bolt is considered ready with status TERMNOK (Terminated NOK) while the remaining bolts continue the terminate sequences.


RUNNING
NORMAL
An error has occurred and there is at least one bolt to terminate
The terminate action is ready and there is no bolt to repair
An error has occurred and there is no bolt to terminate but at least one bolt to repair
A new error occurred and the Failure during repair or Fatal failure settings says there is at least one bolt to terminate
A new error occurred and the Failure during repair settings says there is no bolt to terminate but at least one to repair
RUNNING
REPAIR
RUNNING
TERMINATE
The terminate action is ready and there is at least one bolt to repair
All bolts ready with repair
Evaluation of all bolts using First and Fatal failure gives that there is no bolt to terminate but at least one bolt to repair
Evaluation of all bolts using First and Fatal failure gives that next step in the main program should be started
REPAIR
READY
Evaluation of all bolts using First or Fatal failure gives that there is at least one bolt to terminate

The states of the Bolt

With respect to reject management a bolt is always in one of the following states:

State	Description
Idle	The bolt is done with any previous cycle and is just waiting for a new cycle to start.
Running Normal	The bolt is executing steps in the main part of its program. When it receives a “Start Step” command from the station it executes all steps until the next synchronization mark or that a step fails. “Start Step” with Retry or Terminate order causes a state switch if Retry or Terminate steps are defined.
Running Repair	The bolt is running its repair sequence. The bolt enters this state when it is ordered to start repair. It will then remember the synchronization interval from which it entered this state as the "Failing synchronization interval" (see the picture of the below example). This information is used when the bolt determines if it is ready with the repair The bolt will stay in this state until it again stops in this interval, either due to that it failed to execute one of the steps in the interval or due to that it reached the ending synchronization mark of the interval. **Note!** Reaching any CE step in a program will end the "Failing synchronization interval" and thereby end the repair phase. This regardless if the execution has reached the same CE step that the repair was started from or not. Should the bolt be ordered to enter repair again while already in this state it will remain in this state, without changing the "Failing synchronization interval".
Running Terminate	The bolt is running steps in its terminate sequence.



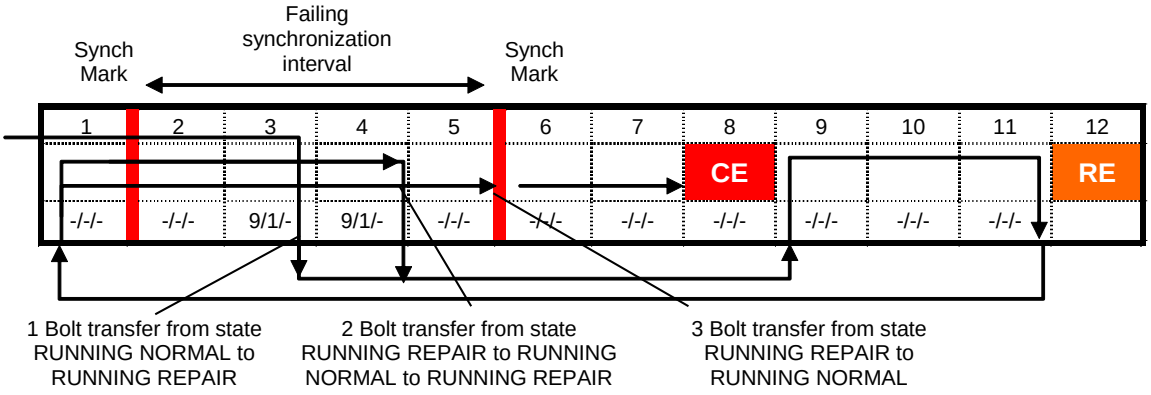
The bolt is ready with repair when:

- The bolt reaches CE step.
- The bolt reaches RE step and no Repair with step is configured.
- All steps from Repair with step to the last step in the failing synchronization interval has been run.

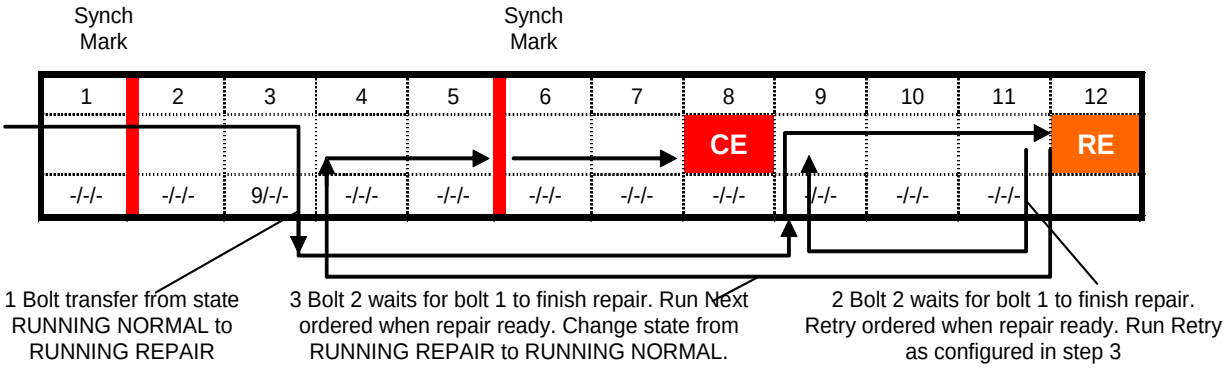
The bolt failed running one of the steps in the failing synchronization interval. This causes an immediate First try evaluation that affects all not terminated bolts in the station. If the evaluated reject action is Retry the bolt reenters Running Repair.

If an initial repair ends with RE step and no repair step is configured cycle execution resumes at the step where the Retry was ordered. Next step when running normal is the step after the step where Retry was ordered. When another bolt fails in its Failing synchronization interval and the action determined by the station is Retry or Terminate the steps run by the OK bolt is configured in the step where the retry was ordered.

See the following example with Bolt 1 and Bolt 2. Bolt 1 fails in step 3 and 4. First failure Retry run on both steps for all bolts:

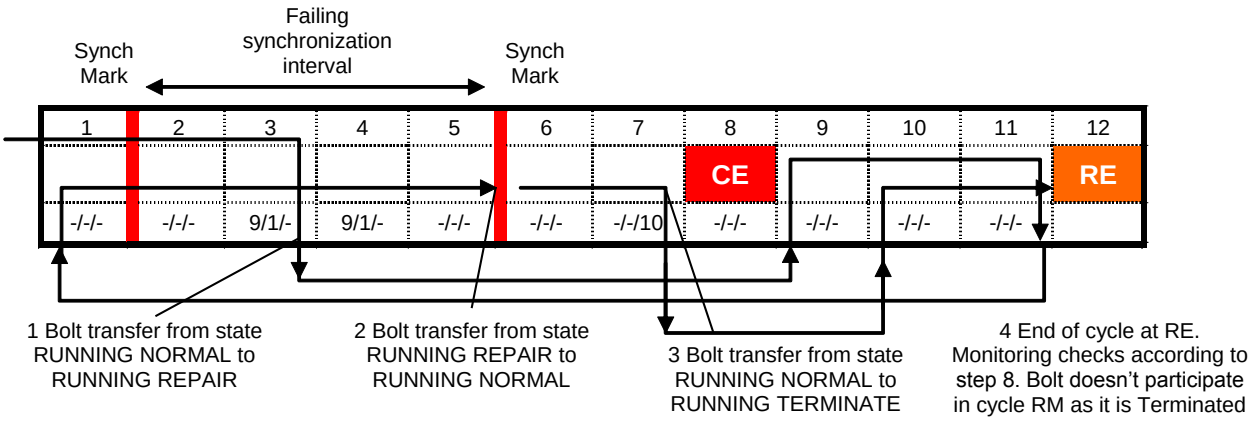


Bolt 2 runs OK, but is ordered to Retry when bolt 1 fails:

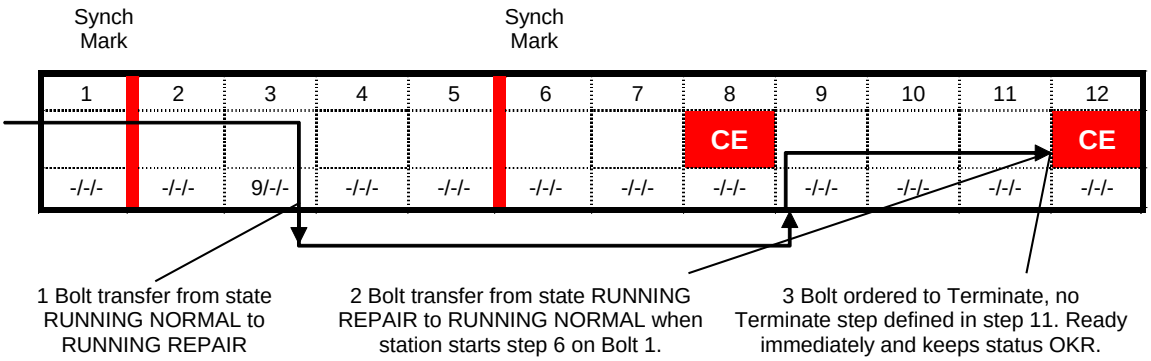


When a bolt has reported that repair is ready, due to a CE step, the bolt will reenter state Running Normal while waiting for the other bolts to finish. If any other bolt fails when running normal, reject management will affect the bolt at CE. Retry and Terminate steps will be run as configured in the last step before the CE step.

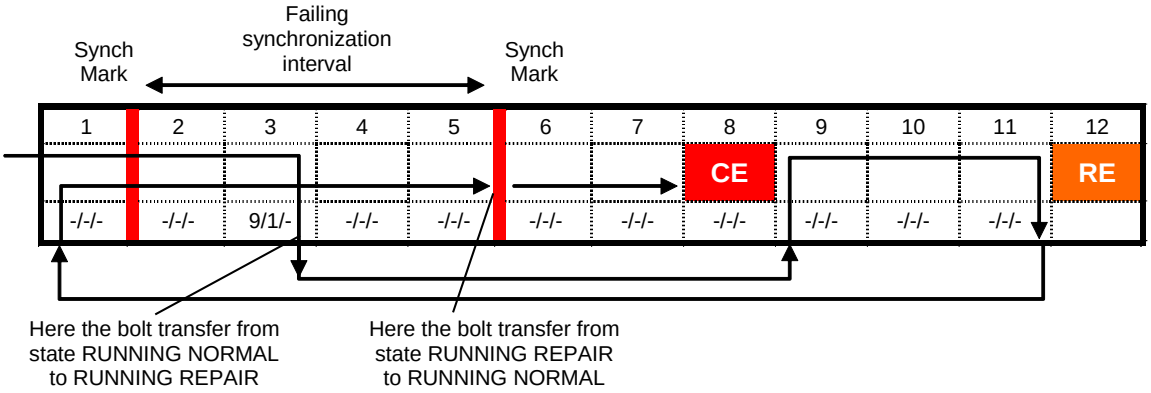
See the following example with Bolt 1 and Bolt 2. Bolt 1 fails in step 3 and 7. In step 3 First failure action is Retry for all bolts. In step 7 First try action is Terminate for all bolts:



Bolt 2 runs OK, but is ordered to Retry and Terminate when bolt 1 fails:



6.5.5.7 Reject management examples



When the bolt fails the first time in step 3 the station will use the First failure settings of this step when computing the reject management action.

In this case a **Loosen with step** sequence is defined (step 9 to 11) so when the station orders the bolt to start repair the bolt will

- Enter the Running Repair state
- And jump to step 9

It will continue to execute without interaction with the station until it reaches a synchronization mark. Here the bolt will stop after step 11 since step 12 is a Return from Repair (which is an implicit synchronization mark). This ends the **Loosen with step** sequence.

Since **Restart from step** is set to 1 for the step 3 the bolt will jump to step 1 when it receives the next start order from the station. Since step 1 has a synchronization mark the bolt will stop here and wait for a new order from the station.

When the station sends the next start order the bolt will run until step 5 is finished. Since the bolt now have reached its "Failing synchronization interval" it transfers from state Running Repair to the Running Normal. The station will not order the bolt to start the next step until all other bolts that also are running repair are finished.

When the next start order is received from the station the bolt will run until it reaches step 7 and at this point report to the station that has finished the program.

When all bolts have finished their cycles the station will order the bolt to do monitoring checks. Monitoring checks will be performed as configured in step 8.

Alternative sequences:

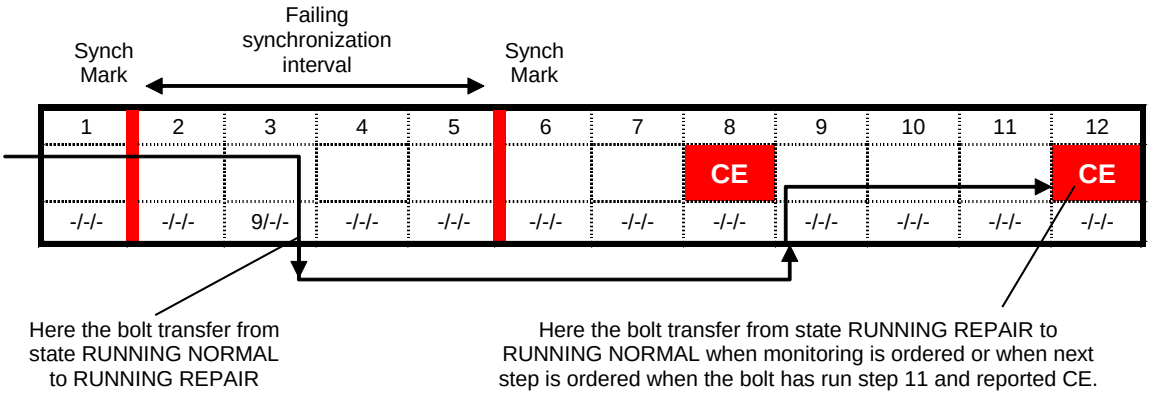
Should the bolt fail while executing step 1 the second time (that is, while Running Repair) it would cause a new reject management action according to the Failure during repair settings of step 1. This since it has not yet reached its "Failing synchronization interval".

Should the bolt fail to execute one of the steps 2, 3, 4 or 5 the second time it will immediately return to the Running Normal state since it reached its "Failing synchronization interval". The station would then wait until all bolts is in the Running Normal state and then compute a new reject action according to the First failure settings of the failing step (2, 3, 4 or 5).

Should the Initiate repair steps end by a CE step the cycle ends at step 11 and monitoring and cycle RM is performed according to step 12. If some monitoring check fails repairable First failure settings for the step are used to determine the action. Fatal errors cause Fatal failure settings to be used.

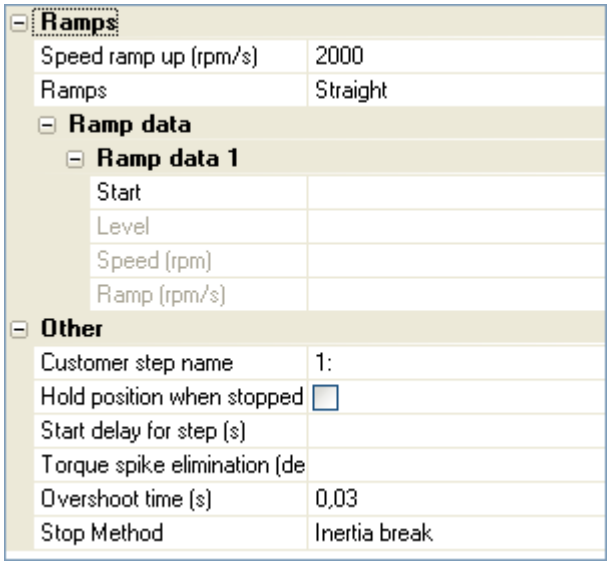
When step 11 has been run the bolt reports step status and that it has reached cycle end and RM end. If there was an error when running step 11 it is handled with Failure during repair settings. If there are other bolts still running when the repair is ready the station will order the bolt at CE too to run next step, upon which the bolt repeats that cycle end is reached and changes state to Running Normal. After this the bolt might be ordered to retry or to terminate, due to other reject bolts.

If some of the bolts in the system run longer cycles, with more synchronization points, than other bolts it is recommended to add a last step before the CE step in the shorter programs. This extra step does not contain any repair steps or terminate steps. It could be for example a “Run until next step” (NS) or a “Wait” (W) step. This saves the shorter cycles from getting involved in reject management actions when they have finished.



6.5.6 Step – Ramps & Other

The Ramps & Other group contains parameters related both to speed ramping of the step (**Ramps**) and other options (**Other**).

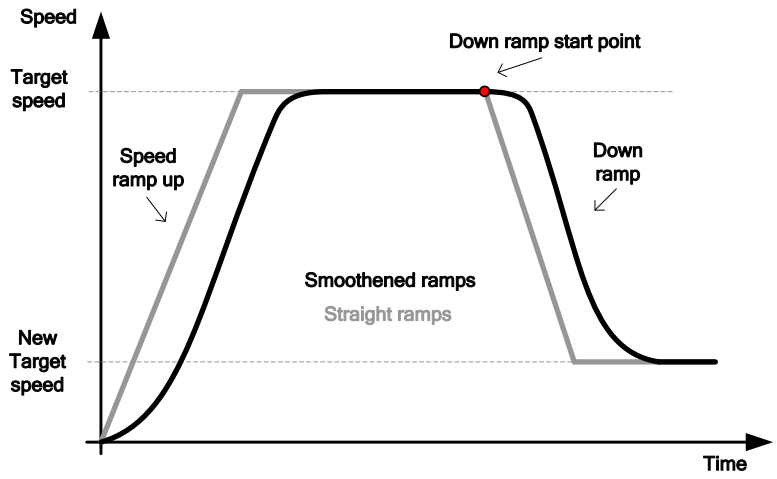


6.5.6.1 Ramps & Other – Ramps

The **Speed ramp up** parameter controls the ramp of the step start and is important to tune to the process in order to avoid unecessary damage and wear of equipment.

The speed ramps defined on a step can be smooth or straight. Straight ramps are defined by a linear equation while smooth follows an S-function. The type of ramp to use is selected in the parameter **Ramps**.

If needed, you can control every aspect of a step's speed curve by use of **Ramp data** parameters.



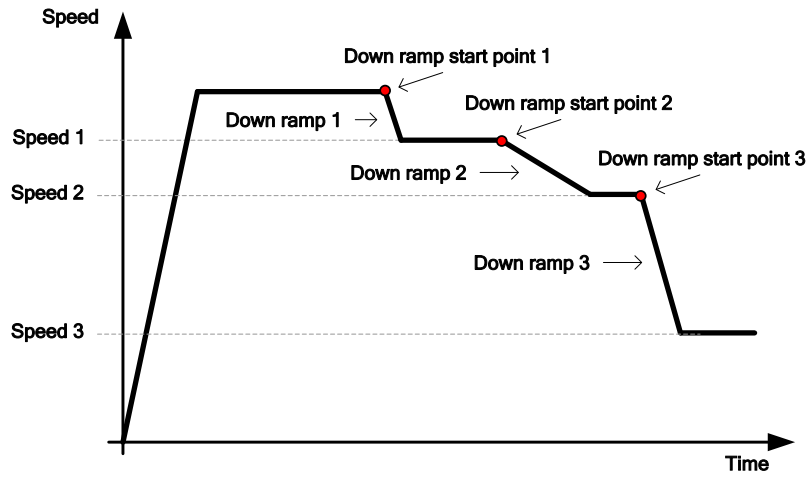
Up to five speed changes can be made during a step. These are defined as **Ramp data** entries.

When to change speed can be triggered by one of the following conditions:

- Angle reaches a specified value
- Torque reaches a specified value
- Current reaches a specified value
- A specified time from step start

Chose type by selecting the wanted type in the **Start** column and specify the trig value in the **Level** property. Enter the new target speed in **Speed** and the acceleration/deceleration to use in **Ramp** [rpm/s].

Only one of the start conditions is monitored at the time. The second condition will not be checked until the first condition has been met, and so on. However, there is no check done that the speed has reached the final value of the first condition before the second is activated.



6.5.6.2 Ramps & Other – Other

The **Other** group holds parameters that do not belong to the earlier parts.

Specify in **Start delay for step** if you want a delay first in the step, before the main function starts.

If the checkbox **Wait for PLC at step end** is checked for at least bolt in the cycle the station will stop and wait for the signal INTERMEDSTART from the PLC to continue running the cycle. This works like an implicit synchronization point. Even if the step is lacking the synchronization point it will still report to the station and wait for all other spindles.

In **Torque Spike Elimination** specify the angle for which all torque readings should be disregarded for control.

Note! The torque restriction (see chapter: Fail Safe Torque) is active also during the **Torque Spike Elimination** phase.

Use the **Customer step name** combo to select the customer error code series that shall be used in this step. See Set Customer step names and Set Customer error codes for more information about this function.

By checking **Hold position when stopped** the bolt will be held in the position it has when zero speed is detected. This option cannot be used together with the **Hold torque** stop condition.

Using the **Stop method** property you can set up how you want to stop the spindle when the target for the step has been reached:

- **Inertia brake**. Choose this alternative if you want to stop as quickly as possible. This is the recommended stop alternative for fastening.
- **Ramp down the current**. First lower the current to X % of the current value at target (**Initial value**), and after that ramp down the current with the **Slope** [A/s]. Use this alternative to relax any torsion built up in a controlled manner.

- **Hold torque**. This alternative does not stop any ongoing motion. The servo will maintain the torque it has when the step is stopped. Note! This means that if there is not contradicting force the spindle will continue to run. This stop condition is used for step types DT - Run until Dyna-Tork^{TM}, DT2 - Run until Dyna-Tork^{TM}, method 2, and DT3 - Run until Dyna-Tork^{TM}, method 3.
- **Continue rotation** (only available for Gauging)
  With this stop condition the spindle will not stop between steps, but instead continue to rotate. During the time between the steps the restrictions **Torque, Remove fastener torque limit** and **Transducer Protection** will still be active. The same limits as in the step before will be used. There will also be a time restriction of 60 seconds for the continue rotation. If any of the limits are exceeded the spindle will be stopped directly and the fatal error **CROT** will be set, together with error-code for the error that stopped the spindle. **Failing Step No** will be set to the step before and the reject settings in this step will be used.

  When the next step starts the speed will be ramped to the target speed of that step with the ramp programmed in this step. If the next step has a different direction compared to the previous step the direction change will be made with a speed ramp. An event will also be generated to warn the user that this has happened.

  If the evaluation of any checks fails for a spindle that run **Continue rotation** the motor will be switched off and the spindle will roll out to a stop. The same will also happen if another spindle fails and starts to run reject management. However, if **Allow hold torque also during RM** is checked in the Station Set Up the spindle will continue to rotate in both these cases.

6.5.6.3 Ramps & Other – Store Position (only available for Gauging)

It is possible to store the current position at the end of each step and later return to this position with the step **Run to Position**.

Set Null position is a combo box with following choices:

- **Keep old value** (default): The null position is not changed
- **Half pos**: The null position is set to the position half way through the step.
- **Full pos**: The null position is set to the current position at the end of the step (after the over shoot time has passed)

Set Half position is a checkbox. If checked the half position will be set to the position half way through the step, otherwise the half position will not be changed. Default value is not checked.

Half position is defined as half the angle from step start (after backlash is done) to step stop (the angle there the braking of the spindle begins).

In **Program Settings-Settings** there is a checkbox **Reset Position**. If checked all four position registers (Null, Half, Peak and Low) will be cleared at cycle start, otherwise the positions stored in a previous cycle will be kept.

Any movement between the programs will be measured and the distance to the position registers will be updated accordingly.

6.5.6.4 Ramps & Other – Signals at step start and end (only available for Gauging)

It is possible to set and reset digital outputs directly from the tightening program. At cycle start all the signals will be cleared, i.e. set low. If a signal is already high and are supposed to be set high from a step, nothing at all will happen, and vice versa if it is already low. A signal used for step start or step end is not allowed to be used in the zones in the same step.

Digital signal at step start:
Output signal specifies the signal to use. With the combobox **Change at** it is possible to select if the signal shall be set directly at step start or when a specific angle has been reached. If the signal shall be set high or low is specified with the combobox **Value.**

Digital signal at step end:
The signal to use is specified with the combobox **Output signal**. The signal is set to the selected **Value** (high or low) when the step ends.

For both step start and step end the following digital signals are avaliable:
Local DO 1, … Local DO 4 (the digital outputs directly on the TC)
PLC_DI_1, … PLC_DI_10 (avalible as inputs in the PLC)

6.5.7 Zones (only available for Gauging)

It is possible to divide a tightening step into several zones. Each zone is specified with a **Start angle** and a **Stop angle**, measured from step start (after the backlash correction is ready). It is possible to have up to 10 zones in one step.

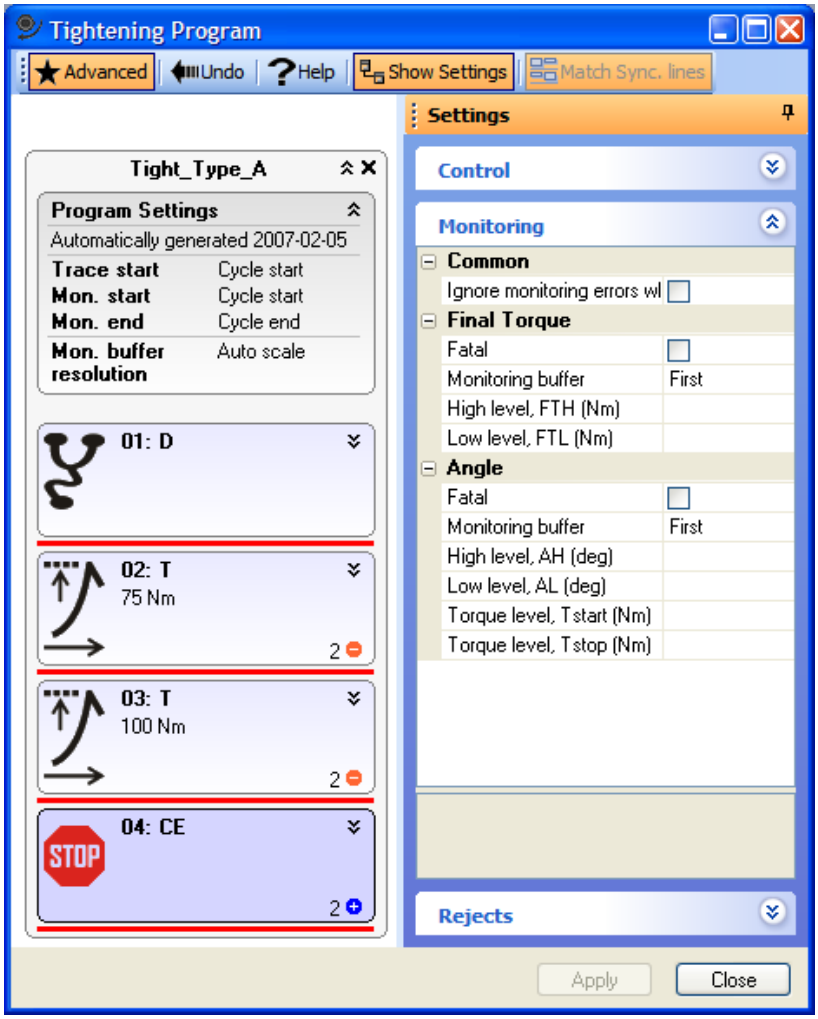
The zones can be used to make a mesurement of some kind during only a part of the step. Each check added to the step can be run during the whole step or only during one specific zone, see Step – Check for more information.

All zones within one step use the same **Angle channel**, as selected in the **Common paramters** for the zones.

It is possible to set a digital signal high at zone start and low at zone end. The signal to use is set with **Output signal**. If **High at start** is checked the signal is set high as soon as the the zone start and if **Low at end** is checked the signal is set low when the zone ends.

The following digital signals are avaliable:
Local DO 1, … Local DO 4 (the digital outputs directly on the TC)
PLC_DI_1, … PLC_DI_10 (avalible as inputs in the PLC)

6.6 Bolt Monitoring



The Monitoring function is used to examine the result of a whole tightening cycle.

This is done using a buffer, called the Monitoring buffer, in which each element represents a given angle interval measured from when monitoring is started. The buffer has a fix length of 8192 elements but its resolution, i.e. the length that each element represents, can be configured by editing the settings for a program.

Note! It is possible to use a second monitoring buffer. To enable the second monitoring buffer check the **Use 2nd monitoring buffer** check box located in general program settings, see chapter: Program Settings. The second monitoring buffer is sampled on the secondary angle and torque channels if equipped.

The monitoring function can be started and stopped at any point during the cycle. When to start and stop monitoring is configured for each program. The following alternatives are available as start conditions:

- At cycle start (default)
- When a specific step is started
- When torque reaches a specified level

The following stop conditions are available:

- At cycle end (default)
- When torque reaches a specified level

Monitoring is performed when a CE-step is reached and the station orders the bolt to do monitoring. If the CE-step has RM defined the cycle might continue after the first monitoring. If so, monitoring is waiting for a new start. The only way to start monitoring after the first CE is to run a step with the **Start/restart monitoring** parameter checked (see Step – Control). When such a step is started monitoring is reset and started. The next time a CE-step is reached monitoring is performed as specified in that step.

Monitoring evaluation is always performed as specified in the different monitoring checks. Should monitoring not have been restarted and the bolt has terminated the monitoring reevaluates the checks using the same measurement values as last evaluation. Thus the reported results will remain and events will be generated if any errors are found. Check **Ignore monitoring errors when step ends NOK** if events and error bits are not wanted during monitoring evaluation after Terminate.

If monitoring was never started or restarted and any step was started (and the bolt is not terminated) the evaluation is done on reset monitoring measurement values, which causes Fatal errors for all checks that have configured limits.

Monitoring can also be stopped when a step fails. To enable this check the **Stop monitoring when step ends NOK** check box on The Tightening Program form, see chapter; : Program Settings. This, together with the use of the Start/restart monitoring parameter for the steps (see Step – Control) can be useful if you want to report the result from the failing tightening.

Normally the first torque and angle transducers are recorded on the first monitoring buffer and the secondary transducers recorded on the second monitoring buffer. If there is only one channel available it is used for both buffers.

Example: The spindle set up contains two enabled torque channels and one enabled angle channel the second monitoring buffer will use torque channel two and angle channel one.

When monitoring is running and the bolt is run in its forward direction, the maximum torque value measured in each angle interval will be stored in the corresponding buffer position.

When the bolt is run in its backward direction (and monitoring is running), the buffer is backed to the position that corresponds to minimum angle reached during the step. The buffer adjustment is done when the next step or monitoring evaluation starts. All values that correspond to the backed distance are erased. Should you pass the point (that is, angle) where monitoring was started, the monitoring buffer will be restarted the next time the bolt is run in its forward direction. This means that you cannot measure a negative angle using the monitoring buffer. (Should you need to measure a negative cycle angle there is

an alternative method that does not use the monitoring buffer. See description of the Angle monitoring check in chapter: Monitoring Check -- Angle .

If the bolt runs forward again the recording is started from the current position.

Steps of the following types are handled specially:

- D - Diagnostic Step. Is always considered running backward.
- E1 - Run Engage method 1. Is always considered running backward.
- GS - Run Gear Shift. Is always considered running backward.
- SR – Socket Release. Nothing is recorded for this step, neither in the forward, nor backward direction.

When monitoring ends two things are done. First, the "Cycle End Torque", used by the Monitoring Check -- Monitoring Yield point torque and angle check, is stored. It is normally set to the torque measured at the shut off point of the last step, or if this step is a SR – Socket Release, the step prior to it. However, if the shut off point (target) is not reached for the last step executed, or if it did run in reverse direction and thereby deleted values from the buffer, then the last torque value stored in the buffer is used as "Cycle End Torque".

Secondly, the complete buffer is filtered for maximum torque for increasing angle, i.e. the following algorithm is applied on the buffer

$$Torque(k) = MAX(Torque(k), Torque(k - 1)), \text{ where } k = [2..8192]$$

It is the filtered buffer that is used for all monitoring checks.

The monitoring checks are added as regular checks on any CE step. The checks are described in detail in the following sub chapters.

If the parameter **Ignore monitoring errors when step ends NOK** (located under Monitoring on the CE step) is checked then the active checks will **not** cause any events or set any error bits, only calculate and report the corresponding result variables, when the cycle is already NOK. It is wise to check this setting to avoid resulting events and error bits caused by the original error.

Note! All limits of the checks are optional. A check having no limit defined will always generate an OK result. This might be useful if its only purpose is to produce a result variable to be reported.

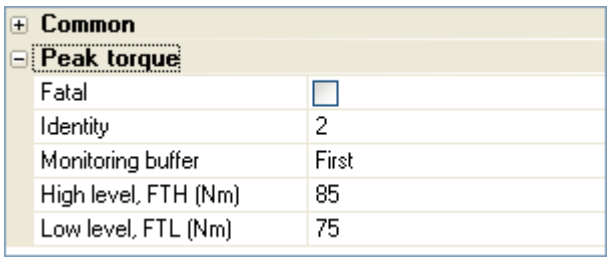
All Monitoring checks can be configured as Fatal or Repairable. This controls which RM action that will be taken.

Note! If the conditions required for evaluation of a check for some reason are not fulfilled an event saying so is generated. Should the check have one or more limits defined then the corresponding error bits are set as well and the status of the checks is set to **Fatal** independent of the configuration. Examples of such conditions are that the buffer is empty, that T start, T stop or threshold torques was not reached, that no yield point was found, etc.

When a monitoring error is found the error is reported as **RM Errors** in cycle data and if not repaired by Cycle RM also as **Errors**. See Cycle RM for details on reject management for monitoring checks.

6.6.1 Monitoring Check – Final Peak Torque

User Interface:



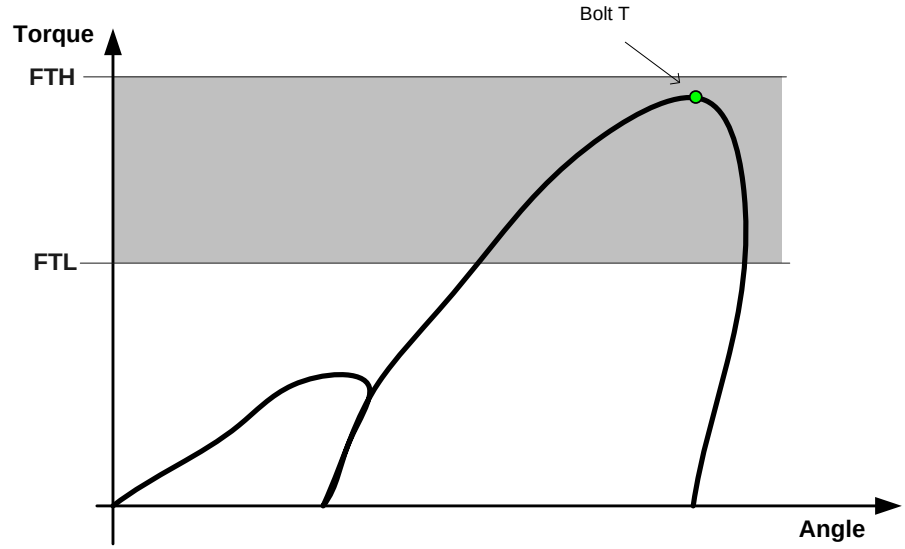
Parameters: **Fatal**, **Monitoring buffer**, **FTH** (High level), **FTL** (Low level)

Function: This function checks that the Peak Torque is within limits **FTH** and **FTL**. The final torque is the highest torque value found in the monitor buffer. The limits **FTH** and **FTL** are both optional. Leaving them blank disables the check but the result value is still calculated and reported.

Unless **Fatal** is checked a value outside the specified limits is considered to be a repairable error. If a value cannot be calculated and at least one limit is specified the check is always considered **Fatal**. This is for example the case if the buffer is empty when evaluating.

Monitoring buffer selects which monitoring buffer the check should use.

Diagram:



FTH	Final Peak Torque High level
FTL	Final Peak Torque Low level

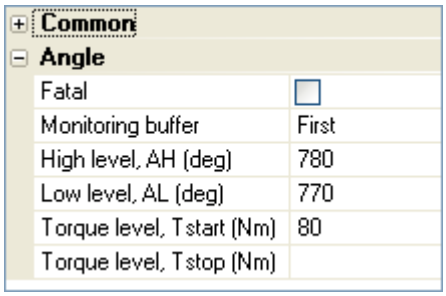
Result var.: "Bolt T" and ”Bolt T2nd” (measured on the second monitoring buffer)

Alarms: Final torque too high (FTHM, FTHM2), Final torque too low (FTLM, FRLM2). FTHM2 and FTLM2 are used for check errors on the second monitoring buffer. If Bolt T is less than the value specified by the Program Settings parameter **Report threshold torque limit** then the warning bit REPLIMIT is set. This does not generate an event unless all bolts that where run during a cycle sets REPLIMIT. If so, the station issues an event (event code 628) to inform that the cycle data was dropped.

Trace: FTH and FTL are shown in trace as horizontal lines from monitoring start to monitoring stop. The lines are only shown when Torque vs. Angle is selected and Monitoring limits are enabled in the trace form (see View of trace curves).

6.6.2 Monitoring Check -- Angle

User Interface:



Parameters: **Fatal**, **Monitoring buffer**, **AH** (High level), **AL** (Low level), **Tstart** (Torque Level, Tstart), **Tstop** (Torque Level, Tstop), **Torque falling below Tstop**.

Function: This function checks that the angle A is within the specified limits **AH** and **AL**. Angle A is measured from the point where the torque has reached **Tstart** (Astart) until the point where the torque reaches **Tstop** (Astop).

Depending on the configuration of **Tstart**, **Tstop** and **Torque falling below Tstop** one of two basically different methods for data collection is used. If **Tstart** and **Tstop** both are left empty, or **Tstop** is specified and **Torque falling below Tstop** is checked, then Astart and Astop are continuously searched for during the cycle. This is the Runtime method. For all other settings the buffer method is used, which means that Astart and Astop are searched for in the buffer.

The Buffer method:
Astart is defined as the angle in the buffer where the torque first exceeds **Tstart**. If **Tstart** is empty Astart is defined as the angle at buffer start. Astop is defined as the angle in the buffer where the torque first exceeds **Tstop** (this is $Astop_1$ in the below picture). If **Tstop** is left empty Astop is defined as the maximum angle reached ($Aend_1$ in the picture).

The Runtime method:

Astart is defined as the angle where **Tstart** is reached for the first time when monitoring is running and values are added to the buffer. If **Tstart** is empty Astart is the cycle angle when monitoring is started (hence it is possible to measure negative angles).

Astop is defined as the angle where torque decreases below **Tstop** ($Astop_2$ in the picture). When torque exceeds **Tstop**, Astop is always reset to an undefined value and a new Astop has to be found for the check to evaluate correctly. An AStop found is valid as long as the torque does not exceed **Tstop** later during the cycle. Found positions are erased and reset if the buffer is cleared passed the found positions (when a step is run in backward direction).

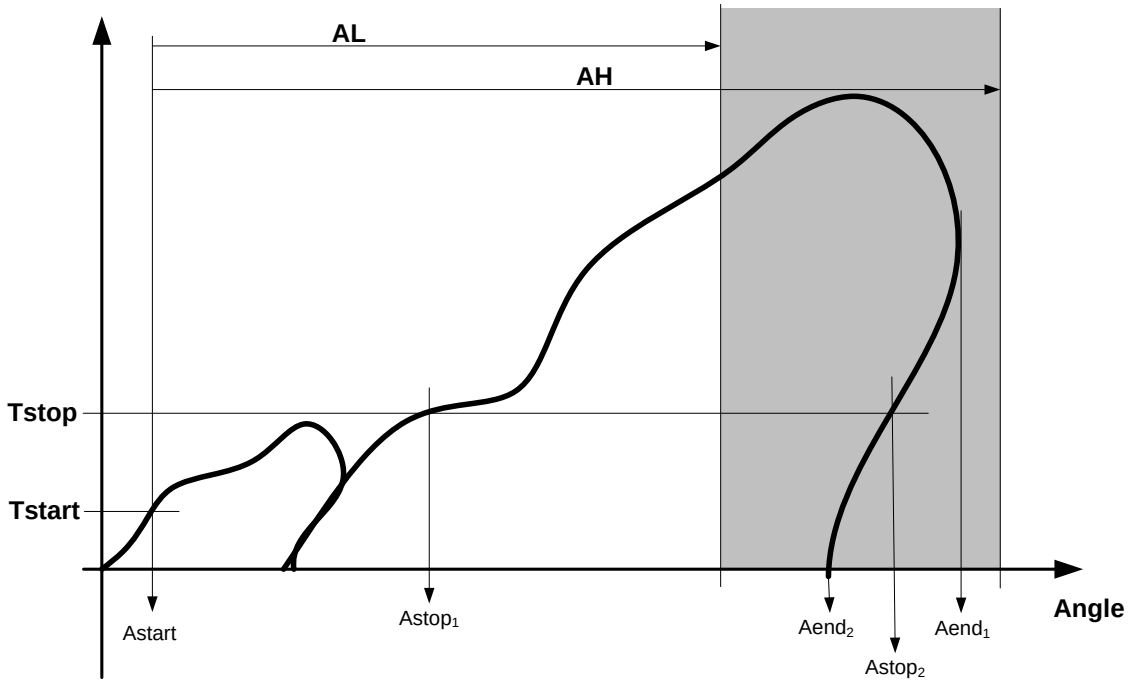
If both **Tstart** and **Tstop** are empty the angle will be measured from monitoring start to monitoring end. That is, Astart is defined as the cycle angle when monitoring is started and Astop is the last measured cycle angle while monitoring is running ($Aend_2$ in the picture). Astop is measured during all steps except for the SR step.

Both limits, that is, **AH** and **AL**, are optional. Leaving them blank disables the check but the result values are still calculated and reported.

If **Fatal** is not checked a value outside the specified limits is considered to be a repairable error. If a value cannot be calculated and at least one limit is specified the check is always considered **Fatal**. This is for example the case if Astart or Astop are not found or if the buffer is empty.

Monitoring buffer selects which monitoring buffer the check should use.

Diagram:



Tstart	Torque Start	**Astart**	Start cond. – Angle at Tstart
Tstop	Torque Stop	**Astop$_1$**	Stop cond. – Angle at Tstop (torque falling below Tstop unchecked)
AL	Angle Low limit	**Astop$_2$**	Stop cond. – Angle at Tstop (torque falling below Tstop checked)
AH	Angle High limit	**Aend$_1$**	Stop cond. – Peak angle
		Aend$_2$	Stop cond. – Last measured angle

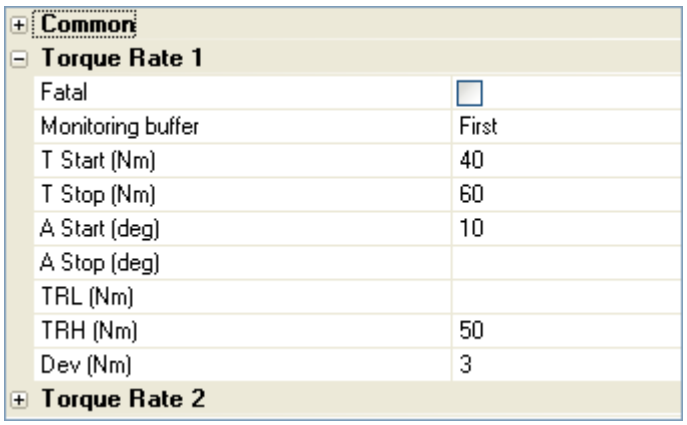
Result var.: "Bolt A", “Bolt A2nd” (measured on the second monitoring buffer) and "Bolt A Thresh T" (the programmed value of **Tstart**)

Alarms: Angle too high (AHM, AHM2), Angle too low (ALM, ALM2) and Threshold torque not reached for angle monitoring (TTNOTRM). AHM2 and ALM2 are used for the second monitoring buffer. TTNOTRM is generated if **Tstart** is specified but not reached during the cycle.

Trace: AH and AL are shown in trace as vertical infinite lines. The lines are only shown when Torque vs. Angle is selected and Monitoring limits are enabled in the trace form (see View of trace curves).

6.6.3 Monitoring Check – Torque Rate

User Interface:



Parameters: **Fatal**, **Monitoring buffer**, **T Start**, **T Stop**, **A Start**, **A Stop**, **TRL**, **TRH**, **Dev**

Function: This function checks the Torque Rate (TR), i.e. the ratio of torque vs. angle.

Measuring is started when the angle **A Start** has elapsed from that torque reached the value **T Start**. This is point (A_1, T_1).

Point (A_2, T_2) is the point where measuring is stopped, that is, when any of the following conditions first is met:

- when the torque reaches the value **T Stop**
- when the angle measured from **T Start** reaches **A Stop**
- when the torque reaches final torque, that is, the end of the monitoring buffer. This point is used only if both **T Stop** and **A Stop** are empty.

T Start, **T Stop**, **A Start**, and **A Stop** are all optional. If **T Start** is left blank, then **A Start** is measured from the start of the monitoring buffer. Leaving **A Start** empty is equal to setting **A Start** to zero. Leaving **T Stop** and/or **A Stop** empty deactivates the respective check as stop conditions.

The Torque Rate is calculated as:

$$TR = (T_2 - T_1) / (A_2 - A_1)$$

where T_1, T_2, A_1, and A_2 are defined as in the diagrams below.

The measured TR is checked to be within the limits **TRH** and **TRL**.

This function also checks that the torque does not deviate more than **Dev** from a straight line between the points (A_1, T_1) and (A_2, T_2). The Deviation in sample k, DEV(k), is

calculated as:

$$DEV(k) = ABS[T(k) – (A(k) * TR + m)]$$

where

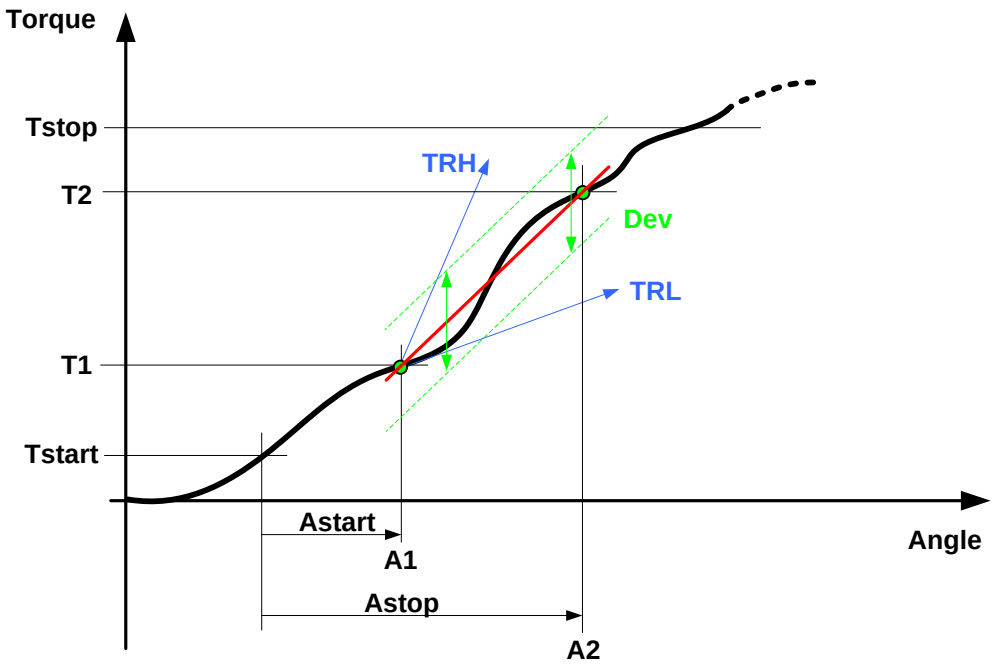
$$m = T_1 – TR * A_1$$

You may specify one or two intervals, which may overlap.

All limits, that is, **TRH**, **TRL** and **Dev**, are optional. Leaving them blank disables the respective check but the values are still calculated and reported.

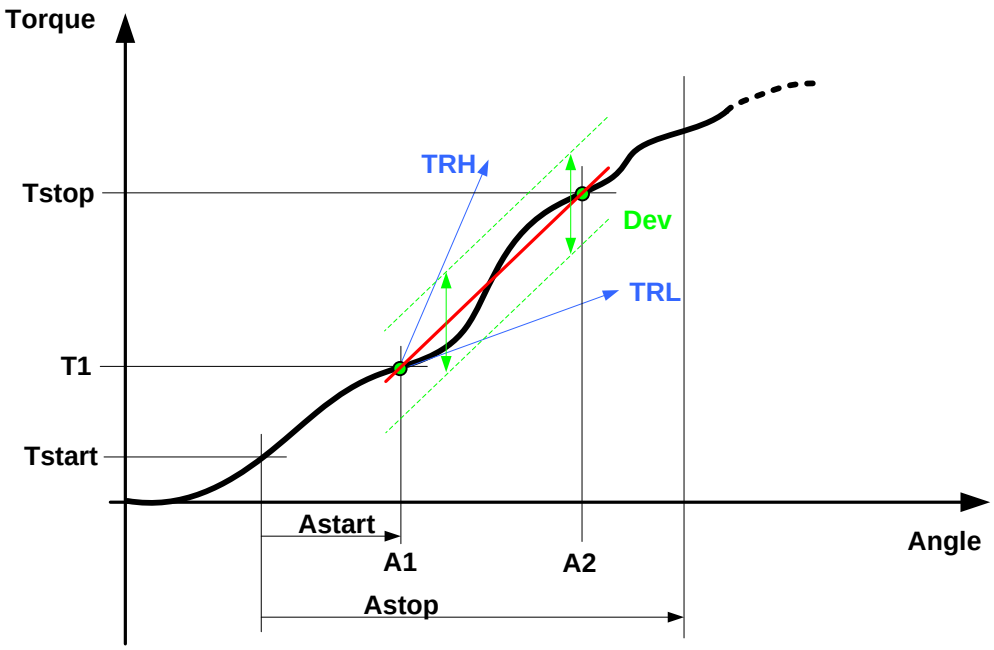
If **Fatal** is not checked a value outside the specified limits is considered to be a repairable error. If a value cannot be calculated and at least one limit is specified for the check in question it is always considered **Fatal**. This is the case if T Start, A Start, T Stop or Astop are not found or if the buffer is empty.

Diagram 1:



Tstart	Torque Start	**Astart**	Angle Start	**Dev**	Deviation
Tstop	Torque Stop	**Astop**	Angel Stop	**TRL**	Torque Rate Low
				TRH	Torque Rate High
T1	Torque at the end of Astart				
T2	Torque at Astop				

Diagram 2:



Tstart	Torque Start	**Astart**	Angle Start	**Dev**	Deviation
Tstop	Torque Stop	**Astop**	Angel Stop	**TRL**	Torque Rate Low
				TRH	Torque Rate High
T1	Torque at the end of Astart				
A2	Angle at Tstop				

Result var.: "Bolt TR 1", "Bolt TR 2", "Bolt TR Dev 1", and "Bolt TR Dev2".

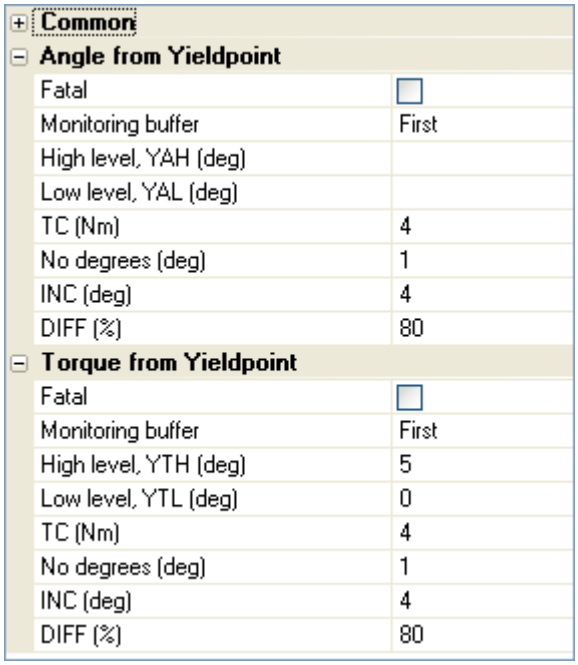
Alarms:
Torque rate in interval 1 too high (TR1HM),
Torque rate in interval 1 too low (TR1LM)
Torque rate in interval 2 too high (TR2HM)
Torque rate in interval 2 too low (TR2LM)
Too big deviation in interval 1 (DEV1M)
Too big deviation in interval 2 (DEV2M)

Trace: TRH and TRL are shown in trace as lines originating from T1 showing the selected torque rate. The TRH and TRL lines end where either T2 or A2 should have been reached using the specified torque rates (see above diagrams). The deviation limit lines are drawn the distance Dev from, and parallel to, the line between (A1, T1) and (A2, T2). The lines are only shown when Torque vs. Angle is selected and Monitoring limits are enabled in the trace form (see View of trace curves).

6.6.4 Monitoring Check -- Monitoring Yield point torque and angle

The monitoring checks Angle from Yieldpoint and Torque from Yieldpoint are described as one item since their configuration is very similar.

User Interface:



Parameters: **TC**, **NOS**, **INC**, **DIFF**, **Torque check Active**, **Torque check Fatal**, **YTH**, **YTL**, **Angle check Active**, **Angle check Fatal**, **YAH**, **YAL**

Function: Search for Yield point starts when the torque value has reached the trig level **TC**. From this point the average torque is measured over **NOS** degrees (T_{NOS}). This procedure is repeated after each **INC** degrees.

When three or more T_{NOS} values have been measure (that is after 3 **NOS** + 2 **INC**) the difference in average torque, $T_{DIFF}(k)$, is calculated as

$$T_{DIFF}(k) = T_{NOS}(k) - T_{NOS}(k - 2)$$

This value corresponds to the current torque rate. Also, every time T_{DIFF} is calculated its maximum value, $MaxT_{DIFF}$, is updated

$$MaxT_{DIFF}(k) = MAX(T_{DIFF}(k), MaxT_{DIFF}(k - 1))$$

The yield point is considered reached when $T_{DIFF}(k) < MaxT_{DIFF}(k) * DIFF / 100$

Two result variables are produced. "Bolt YP T", which is how much the torque increases after the yield point, is calculated as

$$\text{"Bolt YP T"} = T_{\text{Cycle End}} - T_{YP}$$

where $T_{\text{Cycle End}}$ is the Cycle End Torque (as described by Bolt Monitoring).

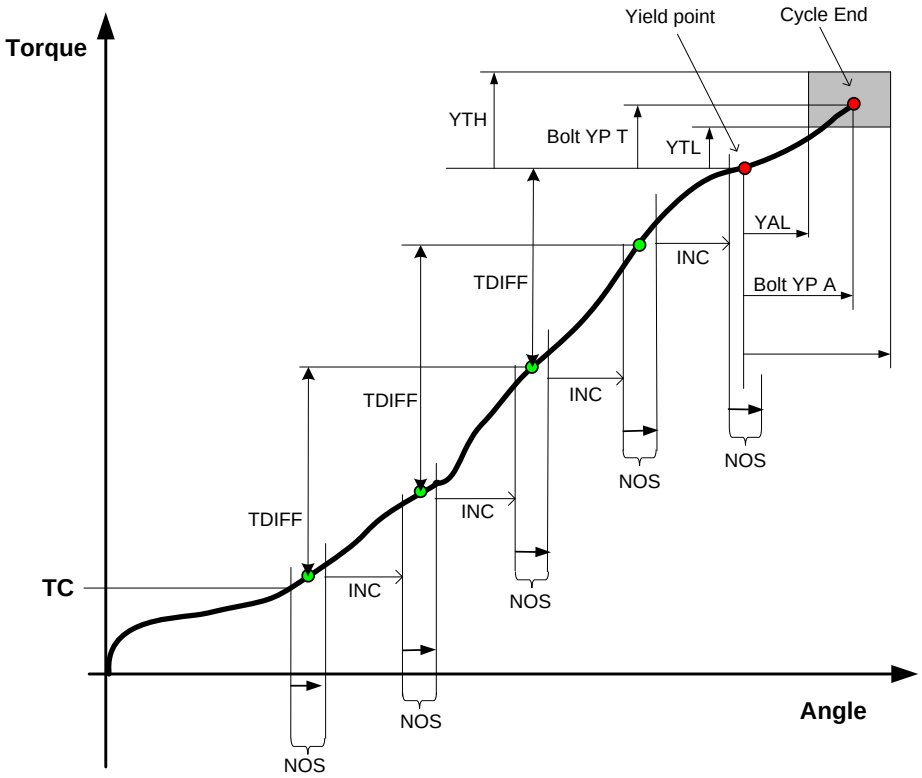
The second variable, "Bolt YP A" represents how much the angle is increased from the yield point to the cycle end. It is calculated as:

$$\text{"Bolt YP A"} = A_{\text{Cycle End}} - A_{YP}$$

The function checks that "Bolt YP T" is within the limits **YTH** and **YTL** and that "Bolt YP A" is within the limits **YAH** and **YAL**. Note that **YTH** and **YTL** can be negative.
All limits, that is, **YTH**, **YTL**, **YAH** and **YAL**, are optional. Leaving them blank disables the respective check but the values are still calculated and reported.

If **Fatal** is not checked a value outside the specified limits is considered to be a repairable error. If a value cannot be calculated and at least one limit is specified for the check in question it is always considered **Fatal**. This is for example the case if the yield points are not found or if the buffer is empty.

Diagram:



TC	Torque TC	**INC**	Increment
NOS	No. degrees	**TDIFF**	Difference in average torque

Result var.: "Bolt YP T" and "Bolt YP A".

Alarms: Yield torque high (YTHM). If "Bolt YP T" > YTH
Yield torque low (YTLM). If "Bolt YP T" < YTL
Yield angle high (YAHM). If "Bolt YP A" > YAH
Yield angle low (YALM). If "Bolt YP A" < YAL

Trace: YAH and YAL are shown in trace as vertical infinite lines. YTH and YTL are shown in trace as horizontal lines from monitoring stat to monitoring stop. The lines are only shown when Torque vs. Angle is selected and Monitoring limits are enabled in the trace form (see View of trace curves).

6.7 Result variables

This chapter describes which system data and tightening results that are available for reporting to the different devices in the system.

What data to report to a particular device is configured using a Reporter. How to create and configure reporters is described in the chapter: **Reporter**.

The result data is divided in the following sections

- Station level data
- Bolt level data
- Step level data

If a result variable for some reason does not have a value, e.g. due to that the check that produces the value is not included in the program, then the value NOT_DEFINED is reported for it.

A NOT_DEFINED value is printed as blanks (spaces) when reported as text and the value –32768 when reported in binary format.

6.7.1 Statuses

Stations and bolts have a variable named "Status" that indicates their resulting status.

The Station Status variable can take one of the following values:

Mnemonic	Num. Id	Description
OK	0	Successful. No repairs have been made.
OKR	1	Successful. Errors have occurred but have been repaired
NOK	2	Not successful. Either a fatal error has occurred or a bolt failed to repair a repairable error.
TERMNOK	3	Not successful. An error occurred while running a terminate sequence (i.e. the steps specified using “Terminate with step” on the Reject tab for a step). The execution of steps was terminated when the error was found

Bolt Status variables can take one of the following values:

Mnemonic	Num. Id	Description
OK	0	Successful. No repairs have been made.
OKR	1	Successful. Errors have occurred but have been repaired
NOK	2	Not successful. This bolt failed to execute a step either due to a restriction, a check or that any other fatal error occurred.
TERMNOK	3	Not successful. An error occurred while running a terminate sequence (i.e. the steps specified using “Terminate with step” on the Reject tab for a step). The execution of steps was terminated when the error was found
NOKRM	4	Not successful **only due to reject management**. The bolt did not fail to execute a step but was ordered to terminate, or not to finish the program, as a reject management action.

6.7.2 Station level result variables

The Station level contains information concerning the station that runs the tightening.

Variable name	Description
Data No	A device unique sequence number. This number is incremented by one for each cycle data reported over the device in question. This number can be used to detect missing cycles.
Station	The name of the station. See System for how to set or change the name of a station. Max. 20 ASCII characters.
Station No	The number of the station. The first station in the system is 1.
Time	The date and time when the cycle was started.
Wp ID	Work piece Identity. Supplied to station from an ID device (see ID device Set Up) or set by the PLC using the IDSTRING output (Station variables). Max. 40 ASCII characters.
Mode	The name of the mode that was executed. Set using The Mode Table form. Max. 20 ASCII characters.
Mode No	The number of the mode that was executed.
Status	Total status for the station: See chapter: Statuses for possible values.
Total	Total number of cycles executed during the current shift. See Shift Reports and Shift Set Up for a description of shifts.
Total OK	Total number of cycles that ended OK or OKR during the current shift. See Shift Reports and Shift Set Up for a description of shifts.
Total NOK	Total number of cycles that ended NOK or TERMNOK during the current shift. See Shift Reports and Shift Set Up for a description of shifts.
Free Str	The value of the PLC output FREESTRING when the cycle was started (see Station variables).
Free No 1	The value of the PLC output FREENUM1 when the cycle was started (see Station variables).

Variable name	Description
Free No 2	The value of the PLC output FREENUM2 when the cycle was started (see Station variables).
Free Str 2	The value of the PLC output FREESTRING2 when the cycle was started (see Station variables).
Free Str 3	The value of the PLC output FREESTRING3 when the cycle was started (see Station variables).
Data No Station	A tightening unique sequence number. This number is incremented by one for each cycle produced by the station.
Data No System	A tightening unique sequence number. This number is incremented by one for each cycle produced by the system.
Station QO	A variable specific for the PLUS protocol. The variable will only have a valid value if a PLUS device is defined. The value is taken from the QO/SA Table in the PLUS device.
Station SA	A variable specific for the PLUS protocol. The variable will only have a valid value if a PLUS device is defined. The value is taken from the QO/SA Table in the PLUS device.
Station AB	A variable specific for the PLUS protocol. The variable will only have a valid value if a PLUS device is defined. The value is taken from Station AB in the PLUS device.
ID Res 1	Multiple identifiers result variable 1. Max. 40 ASCII characters.
ID Res 2	Multiple identifiers result variable 2. Max. 40 ASCII characters.
ID Res 3	Multiple identifiers result variable 3. Max. 40 ASCII characters.
ID Res 4	Multiple identifiers result variable 4. Max. 40 ASCII characters.
No Bolts	Number of bolts in the Cycle Data

6.7.3 Bolt level result variables

Bolt level data is divided in two parts, Common data and Monitoring data where the latter consists of all variables that are produced by the Bolt Monitoring functions.

Common data

The Common data consists of the following variables:

Variable name	Description
Bolt	The name of the bolt. Set System for how to set or change the name of a bolt. Max. 20 ASCII characters.
Bolt No	The number of the bolt as configured.
Spindle No	The number of the TC used to tighten the bolt as configured.
Program	The name of the program used to tighten the bolt. Max. 20 ASCII characters.
OP mode	Operational mode of the bolt. Takes one of the following values: • 0: Connect normally. The bolt was executed during the cycle. • 1: Disconnected with OK status. The bolt was not executed during the cycle. • 2: Disconnected with NOK status. The bolt was not executed during the cycle.
Status	Overall status for the bolt including monitoring. See chapter: Statuses for possible values.
Total	Total number of cycles that this bolt has run during the current shift. See Shift Reports and Shift Set Up for a description of shifts.
Total OK	Total number of OK and OKR cycles that this bolt has run during the current shift. See Shift Reports and Shift Set Up for a description of shifts..
Total NOK	Total number of NOK and TERMNOK cycles that this bolt has run during the current shift. See Shift Reports and Shift Set Up for a description of shifts.
Total Type	The total number of cycles that all bolts, which is of the same type as this bolt, have run during the current shift. The bolt type is a property that can be accessed from the System Map. See Shift Reports and Shift Set Up for a description of shifts.
Total Type OK	The total number of OK and OKR cycles that all bolts, which is of the same type as this bolt, have run during the current shift. The bolt type is defined using the System Map. See Shift Reports and Shift Set Up for a description of shifts.
Total Type NOK	The total number of NOK and TERMNOK cycles that all bolts, which is of the same type as this bolt, have run during the current shift. The bolt type is defined using System Map. See Shift Reports and Shift Set Up for a description of shifts.
Errors	Lists all errors that have occurred during the cycle and is not repaired. See Errors for possible values.
RM Errors	Lists all errors that have occurred during the cycle including all that have been repaired. See Errors for possible values.
Warnings	Lists all warnings that have occurred during the cycle. See Warnings for possible values.

Variable name	Description
Compact Errors	This is a compact version of the "Errors" result variable It may list the following errors: • THM -Torque High Monitoring (bit 0, value 1) • TLM -Torque Low Monitoring (bit 1, value 2) • AHM -Angle High Monitoring (bit 2, value 4) • ALM -Angle Low Monitoring (bit 3, value 8) • OK – Bolt status is OK (bit 4, value 16). This bit is set if the bolts final status is OK. It means that there where no errors at all. When reported binary, each error is represented by the bit written within parentheses above (bit 0 is the least significant).
Failing Step No	The number of the last step that failed in the main part of a program. Only valid if bolt status is NOK, or TERMNOK.
Pgm No	The optional number assigned to the program that was used to tighten the bolt. Set using The Tightening Program form.
Ordinal No	Ordinal Bolt Number. This is always an enumeration of the bolt's position within the station.
Bolt A Thresh T	The threshold torque level Tstart for measuring of Bolt A.
Spindle Serial No	The serial number of the spindle that tightened the bolt.
Compact Errors 2	This is a compact version of the "Errors" result variable It may list the following errors: • TLM -Torque Low Monitoring (bit 0, value 1) • THM -Torque High Monitoring (bit 1, value 2) • ALM -Angle Low Monitoring (bit 2, value 4) • AHM -Angle High Monitoring (bit 3, value 8) • TR1LM - Torque rate in interval 1 too low during monitoring (bit 4, value 16). • TR1HM - Torque rate in interval 1 too high during monitoring (bit 5, value 32). • TR2LM - Torque rate in interval 2 too low during monitoring (bit 6, value 64). • TR2HM - Torque rate in interval 2 too high during monitoring (bit 7, value 128). When reported binary, each error is represented by the bit written within parentheses above (bit 0 is the least significant).
Bolt Pgm Time	The time when the program was last changed or setup downloaded to TC.
Bolt Pgm Strategy	The control strategy of the program. This value is set according to the strategy type of the last step in the program according to rules outlined in chapter: GM DeviceNet/Indication schema. Possible values are: • 2: If the control strategy is **Torque Control**. • 6: If the control strategy is **Angle Control**. • NOT_DEFINED (-32768 if report format is Binary and blank if Binary): If the control strategy is **Undefined**
Spindle Name	Name of the spindle that was used to tighten the bolt.

Variable name	Description
Customer Error Code	A four (4) character string with a customer configurable error code. The mapping of PowerMACS errors to customer error codes are done in the Options window, click on button "Set Customer error codes…"
Mon A Chan	Angle channel used for monitoring.
Mon T Chan	Torque channel used for monitoring.
Con A Chan	Angle channel used for control.
Con T Chan	Torque channel used for control.
Mon Buf 1 A	Angle channel used for first monitoring buffer.
Mon Buf 1 T	Torque channel used for first monitoring buffer.
Mon Buf 2 A	Angle channel used for second monitoring buffer.
Mon Buf 2 T	Torque channel used for second monitoring buffer.
No. Steps	Number of steps stored in the Cycle Data.
Spindle Art. No	Spindle article number
Sp. total cycles	Spindle Total cycles produced by the spindle
Sp. cycles since serv.	Spindle Cycles since last service
Sp. cycl. to serv.	Spindle Cycles to next service
Bolt var. 01..40 T	General cycle data variables for the checks, scaled as Torque if different torque units are selected in reporters.
Bolt var. 41..80	General cycle data variables for the checks

Monitoring data

The Monitoring data contains variables that correspond to the result of the Bolt Monitoring functions.

In addition to the variables listed below the high and low limit set up for it, as well as the Cp, Cpk, and Cam SPC values calculated for them, are also available as result variables.

When including the limits of a variable both the high and the low limit will be included in the report. If you choose to include the SPC values all three values are reported together.

All monitoring data variables are defined once for each monitoring buffer. The variable refering to the second monitoring buffer is appended with “2nd”

Variable name	Description
Bolt T Bolt T 2nd	Peak Torque for the cycle. See Monitoring Check – Final Peak Torque for how it is defined.
Bolt A, Bolt A 2nd	Cycle Angle. Either measured from monitoring start or from when the threshold level Tstart is reached. See Monitoring Check -- Angle for how it is defined.
Bolt TR1 Bolt TR1 2nd	Torque Rate in interval 1. See Monitoring Check – for how it is defined.

Bolt TR2 Bolt TR2 2nd	Torque Rate in interval 2. See Monitoring Check – for how it is defined.
Bolt TR Dev 1 Bolt TR Dev 1 2nd	Maximum torque rate deviation in interval 1. See Monitoring Check – for how it is defined.
Bolt TR Dev 2 Bolt TR Dev 2 2nd	Maximum torque rate deviation in interval 2. See Monitoring Check – for how it is defined.
Bolt YP T Bolt YP T 2nd	Yield Point Torque. See Monitoring Check -- Monitoring Yield point torque and angle for how it is defined.
Bolt YP A Bolt YP A 2nd	Yield Point Angle. See Monitoring Check -- Monitoring Yield point torque and angle for how it is defined.
Bolt Max T	The highest measured torque during the cycle. This value does not use the monitoring buffer and is totally independent from all steps and check, etc. There are no limits for this variable.
Bolt Min T	The lowest measured torque during the cycle. If any loosening takes place during the cycle, this will be a negative value. The value does not use the monitoring buffer and is totally independent from all steps and check, etc. There are no limits for this variable.

6.7.3.1 Errors

The Bolt result variables **Errors** and **RM Errors** can hold the errors listed below.

When reported in binary format the respective error variable occupies 16 bytes (128 bits) where each bit represent an error. The bits are numbered 0 to 127 and are laid out as below:

Byte Offset	0	1	2	3
Error bit no.	31 24	23 16	15 8	7 0

Byte Offset	4	5	6	7
Error bit no	63 56	55 48	47 40	39 32

Byte Offset	8	9	10	11
Error bit no.	95 88	87 80	79 72	71 64

Byte Offset	12	13	14	15
Error bit no	127 120	119 112	111 104	103 96

Note! If Type is set to "I8" then only first 8 bytes are included, that is errors 0 to 63.

Possible errors:

Mnemonic	Bit no.	Description
ANGDIAG	0	Angle count test failed during a diagnostic step
SPFUNCTEST	1	Spindle functional test failed during a diagnostic step
STZODIAG	2	Static zero offset failed during a diagnostic step
DYNZODIAG	3	Dynamic zero offset failed during a diagnostic step
FLYZODIAG	4	Flying zero offset failed during a diagnostic step
TR	5	Fail safe Torque restriction exceeded
AR	6	Fail safe Angle restriction exceeded
TIR	7	Fail safe Time restriction exceeded
TCR	8	Torque – Current restriction exceeded
CROSSTR	9	Cross thread restriction exceeded
CROSSGR	10	Cross gradient restriction exceeded
T1HR	11	Torque in window 1 too high from check
T2HR	12	Torque in window 2 too high from check

Mnemonic	Bit no.	Description
T3HR	13	Torque in window 3 too high from check
T1LR	14	Torque in window 1 too low from check
T2LR	15	Torque in window 2 too low from check
T3LR	16	Torque in window 3 too low from check
PTH	17	Peak torque high alarm from check
PTL	18	Peak torque low alarm from check
AWINTH	19	Torque high in angle window alarm from check
AWINTL	20	Torque low in angle window alarm from check
TIWINTH	21	Torque high in time window alarm from check
TIWINTL	22	Torque low in time window alarm from check
MTH	23	Mean torque high alarm from check
MTL	24	Mean torque low alarm from check
AH	25	Angle high alarm from check
AL	26	Angle low alarm from check
TIH	27	Time high alarm from check
TIL	28	Time low alarm from check
CH	29	Current high alarm from check
CL	30	Current low alarm from check
SOTH	31	Shut off torque high from check
SOTL	32	Shut off torque low from check
T/IH	33	Torque/T3 (Current as Torque) too high from check
T/IL	34	Torque/T3 (Current as Torque) too low from check
PVTH	35	Post view torque too high from check
PVTL	36	Post view torque too low from check
THM	37	Final torque too high during monitoring
TLM	38	Final torque too low during monitoring
AHM	39	Angle too high during monitoring
ALM	40	Angle too low during monitoring
TR1HM	41	Torque rate in interval 1 too high during monitoring
TR1LM	42	Torque rate in interval 1 too low during monitoring
TR2HM	43	Torque rate in interval 2 too high during monitoring
TR2LM	44	Torque rate in interval 2 too low during monitoring
DEV1M	45	Deviation in interval 1 too high during monitoring
DEV2M	46	Deviation in interval 2 too high during monitoring

Mnemonic	Bit no.	Description
YTHM	47	Yield point torque high during monitoring
YTLM	48	Yield point torque low during monitoring
YAHM	49	Yield point angle high during monitoring
YALM	50	Yield point angle low during monitoring
TDIFF	51	Double Torque transducer error
ADIFF	52	Double Angle encoder error
BUFOVFLM	53	Monitoring: Overflow in recording buffer. See description of parameter "Allow monitoring buffer overrun" in Program Settings.
–	54 – 61	Reserved
SAT	62	A/D converter used for torque measurement saturated
OTHER	63	This is bit is set if any error bit(s) in the range 64 . 95 are set and only the first 8 bytes are included. That is, Type in the reporter is set to "I8".
LOOSEN	64	The automatic loosening, defined using the "Run reverse before retry" parameters on the Step – Rejects tab, failed.
TRH	65	Torque rate too high from check
TRL	66	Torque rate too low from check
DEV	67	Deviation too high from check
TTNOTRM	68	Threshold torque not reached for angle monitoring (see also chapter: Monitoring Check -- Angle)
RFTLIMRE	69	Remove fastener torque limit reached (see also chapter: Program)
–	70 – 73	Reserved
PH	74	Absolute Position check high limit
PL	75	Absolute Position check low limit
–	76 – 77	Reserved
ESTOP	78	Emergency stop
MSTOP	79	Machine stop
SSTOP	80	Station stop (Disconnect, problem reported from station to all bolts, connected bolts get Station stop)
DETACH	81	Detach, TC detach detected and reported by station
SERVO	82	Servo problem
DISNOK	83	Bolt is in operational mode Disconnected NOK
TPROT	84	Transducer protection, none configurable restriction active during all steps in cycle
THM2	85	Final torque too high during monitoring on second buffer

Mnemonic	Bit no.	Description
TLM2	86	Final torque too low during monitoring on second buffer
AHM2	87	Angle too high during monitoring on second buffer
ALM2	88	Angle too low during monitoring on second buffer
BUFOVFLM2	89	Monitoring: Overflow in second recording buffer. See description of parameter "Allow monitoring buffer overrun" in Program Settings.
MTR	90	Min Torque restriction exceeded
GHR	91	Gradient restriction, high limit exceeded
GLR	92	Gradient restriction, low limit exceeded
YAH	93	Yield angle, high limit exceeded
YAL	94	Yield angle, low limit exceeded
OTHER2	95	This is bit is set if any error bit(s) in the range 96 . 127 are set and only the first 12 bytes are included. That is, Type in the reporter is set to "I12".
SS	96	Slip Stick Error
SPINDLE	97	Spindle Error
CONFIG	98	Configuration error
DTH	99	Delta torque high alarm from check
DTL	100	Delta torque low alarm from check
CROT	101	Continue rotation
LTH	102	Low spot torque high alarm from check
LTL	103	Low spot torque low alarm from check

6.7.3.2 Warnings

The warning codes displayed in the bolt data table are listed and explained below:

The Bolt result variables **Warnings** can hold the warnings listed below.

When reported in binary format the respective error variable occupies 8 bytes (64 bits) where each bit represent a warning. The bits are numbered 0 to 63 and are laid out as below:

Byte Offset	0	1	2	3
Error bit no.	31 24	23 16	15 8	7 0

Byte Offset	4	5	6	7
Error bit no	63 56	55 48	47 40	39 32

Possible warnings:

Warning	**Bit no.**	**Description**
NZS	0	Not zero speed
FLYZODIAG	1	Flying zero offset failed during a diagnostic step
BUFOVFLM	2	Monitoring: Overflow in recording buffer. See description of parameter "Allow monitoring buffer overrun" in Program Settings.
REPLIMIT	3	Report limit not reached. Monitoring torque (Bolt T) is less than the value specified by the parameter **Report threshold torque limit** (see chapter: Program).
BUFOVFLM2	4	Monitoring: Overflow in second recording buffer. See description of parameter "Allow monitoring buffer overrun" in Program Settings.
SPSERV	5	Time to service spindle
-	5 - 63	Reserved.

6.7.4 Step level result variables

The Step level data is the data that can be reported for each individual step in a tightening cycle. It is divided in two parts, Common data and Check data, where the first covers all variables that available for all steps and the latter all variables produced by checks executed for the step.

Common data

This section lists all variables that are available for all steps without any specific configuration.

Variable name	Description
Step No	The step number. This is the order of the step within the program. First step is one (1). Steps within a sequence are given the step number P * 100 + S where P is the step number of the SE step that calls the sequence and S is the step number within the sequence.
Step Type	The numerical representation of the type of step. See the definition of the respective step for valid values (Step – Control).
Par	The parameters of the steps control part. See the definition of the respective step type (Step – Control) for which parameters that are printed, and in which order. Note This variable always prints 10 values. If fewer parameters are used for a step then the unused will have the value NOT_DEFINED.
Speed	The target Speed for the step.
A Chan Con	The angle channel used for step control
T Chan Con	The torque channel used for step control
Errors	Step errors that have occurred, see Errors.
Step Name	Step name

Check data

The check data consists of the variables that are produced as a result of running a check, a restriction or produced by the steps control function.

In addition to the variables listed below the high and how limit set up for it, as well as the Cp, Cpk, and Cam SPC values calculated for them, are also available as result variables.

Each check data variable is defined once for each possible choice for check channel. The first variable is reported when the default check channel is used (Monitoring).

Variable name	Description
Peak T Peak T1 Peak T2 Peak T3	The highest torque reached during the step. Produced by the "Check peak torque" check.

Variable name	Description
Mean T Mean T1 Mean T2 Mean T3	Mean torque during end of step. Produced by the "Check mean torque" check.
DT Mean T	The mean torque measured during the time interval TI2 of the DT - Run until Dyna-Tork™, DT2 - Run until Dyna-Tork™, method 2, and DT3 - Run until Dyna-Tork™, method 3 steps. Value is based on control channel used for step.
DT T	The torque measured when a DT - Run until Dyna-Tork™, DT2 - Run until Dyna-Tork™, method 2, or a DT3 - Run until Dyna-Tork™, method 3 step switches to using the DT torque as target.
A Win Hi T A Win Hi T1 A Win Hi T2 A Win Hi T3	The highest torque measured in the angle window. Produced by the "Check torque in angle window" check.
A Win Lo T A Win Lo T1 A Win Lo T2 A Win Lo T3	The lowest torque measured in the angle window. Produced by the "Check torque in angle window" check.
Ti Win Hi T Ti Win Hi T1 Ti Win Hi T2 Ti Win Hi T3	The highest torque measured in the time window. Produced by the "Check torque in time window" check.
Ti Win Lo T Ti Win Lo T1 Ti Win Lo T2 Ti Win Lo T3	The lowest torque measured during time window. Measured by “Check torque in time window”.
A A1 A2	The step angle. Produced by the "Check angle" check.
Time	The time that the step was run. Measured from that the servo is started until it is stopped. Produced by the "Check time" check.
T / T3 T1 / T3 T2 / T3	The mean value of torque vs. the mean value of current both measured during the last T s of the step. Produced by the "Check T/T3 (Current as Torque)" check.
Relax A	The angle measured from that target is reached until the step is shut off. Only produced by the DT - Run until Dyna-Tork™, DT2 - Run until Dyna-Tork™, method 2, and DT3 - Run until Dyna-Tork™, method 3 steps. This value is based on the control channel used for the step.
Release A	The release angle measured from the negative peak torque until the point where the line, trough the peak torque point and the shut off point, cuts the angle axis. Is produced by the “LT - Loosen until torque” and the “”RA - Release angle steps. This value is based on the control channel used for the step.

Variable name	Description
Tp1 Peak T Tp1 Peak T1 Tp1 Peak T2 Tp1 Peak T3	The peak torque measured in angle window 1 of the restriction Torque Profile.
Tp2 Peak T Tp2 Peak T1 Tp2 Peak T2 Tp2 Peak T3	The peak torque measured in angle window 2 of the restriction Torque Profile.
Tp3 Peak T Tp3 Peak T1 Tp3 Peak T2 Tp3 Peak T3	The peak torque measured in angle window 3 of the restriction Torque Profile.
Yieldpoint A Yieldpoint A1 Yieldpoint A2	The Yield point angle measured by the Check Yield Point Angle check.
ShutOff T ShutOff T1 ShutOff T2 ShutOff T3	The torque measured at the shut off point of the step. Produced by the "Check shut off torque" check.
TorqueRate T TorqueRate T1 TorqueRate T2 TorqueRate T3	Torque Rate. Produced by the "Check torque rate" check.
TorqueRate Dev TorqueRate Dev T1 TorqueRate Dev T2 TorqueRate Dev T3	Maximum torque rate deviation. Produced by the "Check torque rate" check.
Gradient Gradient T1 Gradient T2 Gradient T3	The gradient measured by the Gradient restriction.
Delta T Delta T2 Delta T2	The difference between the lowest and the highest torque. Produced by the Check delta torque check. (Only available for Gauging)
BCT A BCT A1 BCT A2	Reports measured backlash angle for BCT – Backlash correction (only available for Gauging). (Only available for Gauging)
Zero Offset T1 Zero Offset T2	Reports the last measured Zero Offset of the torque transducer. Measured by the spindle during Spindle functional test in a D - Diagnostic Step.
ZO Drift T1 ZO Drift T2	Reports the Zero offset drift for the torque transducer, that is the difference between the this Zero offset measurement and the previouse one. Measured by the spindle during Spindle functional test in a D - Diagnostic Step.
Shunt Calib T1 Shunt Calib T2	Reports the last measured Shunt calibration value of the torque transducer. Measured by the spindle during Spindle functional test in a D - Diagnostic Step.

Variable name	Description
Shunt Drift T1 Shunt Drift T2	Reports the Shunt claibration drift for the torque transducer, that is the difference between the this Shunt calibration measurement and the previouse one. Measured by the spindle during Spindle functional test in a D - Diagnostic Step.
Position A	The result of the check Check Position (only available for Gauging)
Low Spot T Low Spot T1 Low Spot T2 Low Spot T3	The lowest torque. Produced by the Check low spot torque check (only available for Gauging)
Null Pos A	The distance to the Null position after the current step (only available for Gauging).
Half Pos A	The distance to the Half position after the current step (only available for Gauging).
Peak Pos A	The distance to the Peak position after the current step (only available for Gauging).
Low Pos A	The distance to the Low position after the current step (only available for Gauging).
Step var. 1..40 T	General cycle data variables for the checks, scaled as Torque if different torque units are selected in reporters.
Step var. 41..80	General cycle data variables for the checks

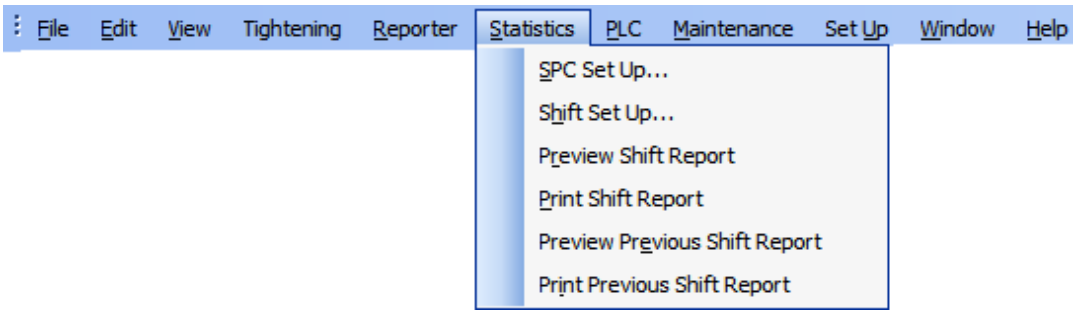
When including the limits of a variable both the high and the low limit will be included in the report. If you choose to include the SPC values all three values are reported together.

7 SPC and Statistics

7.1 SPC - Overview

This part describes how to set up the SPC, Statistical Process Control, and other statistics.

A first introduction to SPC is given in the next chapter:, SPC, Statistical Process Control.



SPC Set Up… opens the SPC set up form with which the SPC is configured.

Shift Set Up… opens Shift Set Up form. Use it to configure shifts.

The other alternatives on the SPC menu are used for printing so-called shift reports. See Shift Reports and Shift Set Up for a description of these.

7.2 SPC, Statistical Process Control

The purpose of the built-in SPC function is to provide the operator or quality control staff with data that will enable them to judge the stability and capability of the assembly process according to standard SPC rules. By automating the SPC function within the PowerMACS the work will be simplified and carried out without the necessity of external SPC charts.

Since any assembly parameter within PowerMACS can be used as a SPC variable, it is necessary to understand that only variables controlled by the PowerMACS reflects the performance of the PowerMACS system. Other variables will mainly reflect results from other processes. If e.g. a torque value is applied to a joint, then obviously SPC of the final torque value, and conclusions from that, such as Cp and Cpk will reflect the performance of the tool and the controller. A SPC study of the angle from some torque level will in the same case reflect mainly other aspects outside the PowerMACS sphere of influence, such as friction of the threaded surfaces, machining of thread and mating surfaces etc. even though it is reported by PowerMACS.

The built-in SPC is very flexible and it is possible to tailor data collection, calculations, and checks to suit most needs.

7.2.1 Data Collection

Data is collected only for those variables that have been set up for collection. Data are collected in two main ways:

- As subgroups for the SPC and TDA
- As latest results, or Short Term Trend

For the **SPC and TDA function** data is collected into subgroups. Only data with bolt status OK are calculated on, results from NOK cycles are ignored. The **Subgroup size** defines how many samples a subgroup is calculated from.

When **Subgroup size** values have been collected they are used to calculate the Average value and the Range or Standard deviation. These values are saved for the subgroup while the original values are lost.

This is repeated with a selectable frequency, either specified in number of samples (**Samples**), or in time between start of subgroups (**Minutes**). Note that the selected frequency only controls when collection of data for a subgroup should be started. When started, the result of each OK cycle is taken until the subgroup is filled.

If frequency is specified as **Samples**, say N, then the collection of a new subgroup is started N samples after that the previous subgroup was started. When started, the result of each OK cycle is taken until the subgroup is filled.

If frequency is specified as **Minutes**, say T, then the collection of a new subgroup is started T minutes after that the collection of the previous subgroup was started. Also, if a subgroup is not complete within 30 minutes it is discarded.

The SPC and TDA function will store and calculate data on **No. of subgroups to save** subgroups. It will not display any values until at least **No. of subgroups to calculate** has been collected. However, from that point it will use all subgroups stored in memory in the calculations.

The short **Term Trend function** is used to give a detailed view of the latest recorded samples. It uses a buffer where the last (**Subgroup size** * **No. of subgroups to save)** samples are stored. Here are values from both OK and NOK cycles collected.

When the Short Term Trend is to be displayed then average and the range or deviation values are calculated for subgroups built from the data in the short Term trend buffer, starting with most resent value.

Which variables to collect are set up uniquely for each bolt/spindle and program.

7.2.2 Calculations for subgroups

Data in a subgroup is calculated as:

$$Average = \overline{X} = \frac{\sum X_i}{n} \; ; i = 1..n$$

$$Range = R = \max(X_i) - \min(X_i) \; ; i = 1..n$$

$$Standard\ deviation = S = \sqrt{\frac{\sum (X_i - \overline{X})^2}{n-1}} \; ; i = 1..n$$

$$S\ is\ approximated\ with = \sqrt{\frac{\sum X_i^2 - \frac{(\sum X_i)^2}{n}}{n-1}} \; ; i = 1..n$$

where n = size of subgroup

7.2.3 Calculations and checks

When specified number of subgroups to use for calculations has been collected, the SPC function will start calculations to determine the statistical stability. This will be repeated every time a new subgroup is ready.

Calculations are performed in following steps: (m = number of subgroups)

Calculation of Average of Average values:

$$\overline{\overline{X}} = \frac{\sum \overline{X}_i}{m} \; ; i = 1..m$$

Calculation of Average of Range or Standard deviation:

$$\overline{R} = \frac{\sum R_i}{m} \; ; \; i = 1..m$$

and

$$\overline{S} = \frac{\sum S_i}{m} \; ; \; i = 1..m$$

Calculation of Control levels (if enabled):

$$UCL_X = \overline{\overline{X}} + A_2 * \overline{R}$$

$$LCL_X = \overline{\overline{X}} - A_2 * \overline{R}$$

$$UCL_R = D_4 * \overline{R}$$

$$LCL_R = D_3 * \overline{R}$$

or

$$UCL_X = \overline{\overline{X}} + A_3 * \overline{S}$$

$$LCL_X = \overline{\overline{X}} - A_3 * \overline{S}$$

$$UCL_S = B_4 * \overline{S}$$

$$LCL_S = B_3 * \overline{S}$$

A_2, D_4, D_3 *and* A_3, B_4, B_3*: see table*

Check and alarm (if enabled) if following happens with Average or Range or Standard deviation:

- one value outside control levels (One out)
- points consecutively increasing (Seven up)
- 7 points consecutively decreasing (Seven down)
- 7 points consecutively above average (Seven above)
- 7 points consecutively below average (Seven below)
- >90% in mid third (Mid gather)
- <40 in mid third (End gather)

Calculation of the process capabilities Cp and Cpk:

$$\sigma_i = \frac{\overline{R}}{d_{5*}}$$

$$d_{5*} = d_5 - 1{,}645 \cdot \frac{0{,}864}{\sqrt{k}}$$

$$s' = \frac{\overline{R}}{d_2}; i = 1..m(d_2 : see\ table)$$

or

$$s' = \frac{\overline{S}}{c_4}; i = 1..m(c_4 : see\ table)$$

$$Cp = \frac{UTL - LTL}{(su + sl) * s'}$$

$$Cpk = \min\left[\frac{UTL - \overline{\overline{X}}}{su * s'}, \frac{\overline{\overline{X}} - LTL}{sl * s'}\right]$$

$$Cam = \frac{UTL - LTL}{(su + sl) * \sigma_i}$$

where

- for a normal distribution su = sl = 3
- d5 = 2.326 for subgroup size = 5
- k = no of subgroups to calculate

Cam is only calculated for subgroup size = 5, and requires range to be collected. Cam is not applicable to standard deviation.

An error event is generated if the value of Cam, Cp or Cpk is out of limit.

When SPC is wanted for a station, not for its individual bolts, the calculations are made on those variables that happen to be set up for data collection on that station.

Station, Program, Step and Variable are still defined, so only variables matching those criteria are used.

It is not possible to specify alarm limits or control limits for SPC calculated on a station.The weighing together of values for the separate bolts are done for each subgroup according to the following rules:

$NoBolt$ = The number of bolts in this station for which SPC - collection is set up.

$\overline{X}_{ij}$ = The mean value for subgroup j for bolt i.

$\overline{XStn}_j$ = The average for the station for subgroup j

Var_{ij} = The varians for subgroup j for bolt i

$VarStn_j$ = The varians for the station for subgroup j

$StdDevStn_j$ = The standard deviation for the station for subgroup j

$$\overline{XStn}_j = \frac{\sum_{i=1}^{NoBolts} \overline{X}_{ij}}{NoBolts}$$

$$VarStn_j = \sum_{i=1}^{NoBolts} \left(\overline{X}_{ij} - \overline{XStn}_j \right)^2 \cdot Var_{ij}$$

$$StdDevStn_j = \sqrt{VarStn_j}$$

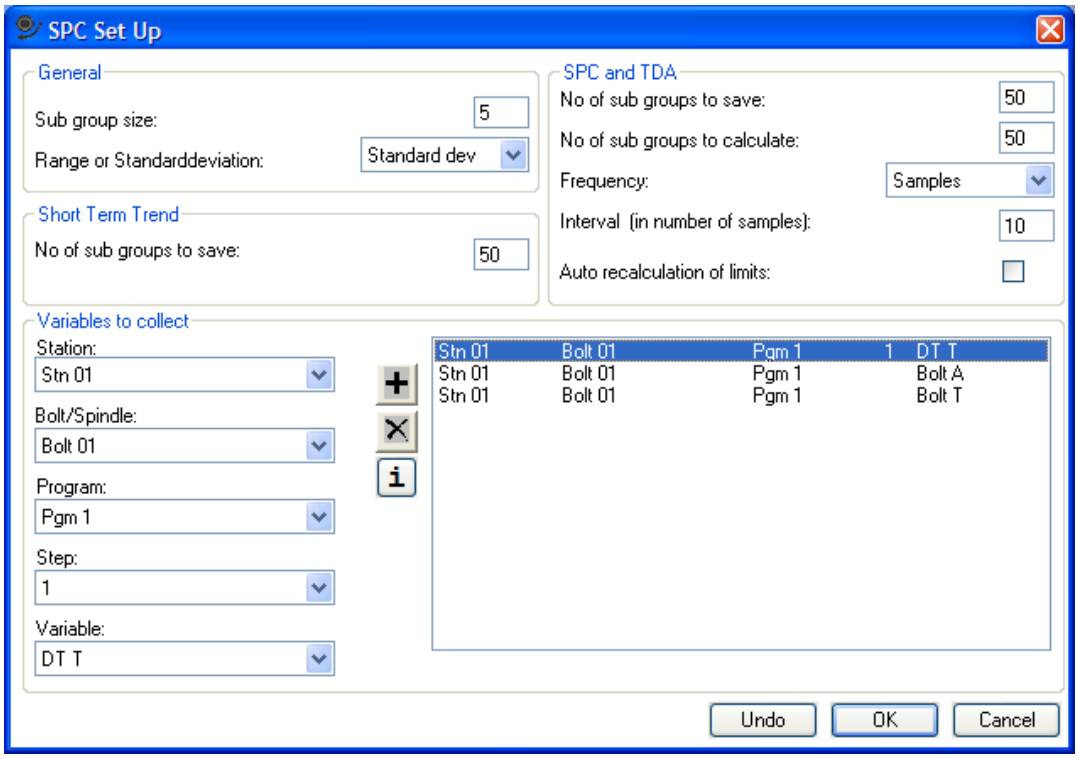
7.2.4 SPC constants

The table below gives the constants to use in the mathematical formulas.

Sub-group size	Divisors for estimate of standard dev.		Factors for Control Limits					
n	d2	c4	A2	D3	D4	A3	B3	B4
2	1.13	0.798	1.88	-	3.27	2.66	-	3.27
3	1.69	0.886	1.02	-	2.57	1.95	-	2.57
4	2.06	0.921	0.73	-	2.28	1.63	-	2.27
5	2.33	0.940	0.58	-	2.11	1.43	-	2.09
6	2.53	0.952	0.48	-	2.00	1.29	0.03	1.97
7	2.70	0.959	0.42	0.08	1.92	1.18	0.12	1.88
8	2.85	0.965	0.37	0.14	1.86	1.10	0.19	1.82
9	2.97	0.969	0.34	0.18	1.82	1.03	0.24	1.76
10	3.08	0.973	0.31	0.22	1.78	0.98	0.28	1.72
11	3.17	0.975	0.29	0.26	1.74	0.93	0.32	1.68
12	3.26	0.978	0.27	0.28	1.72	0.89	0.35	1.65
13	3.34	0.979	0.25	0.31	1.69	0.85	0.38	1.62
14	3.41	0.981	0.24	0.33	1.67	0.82	0.41	1.59
15	3.47	0.982	0.22	0.35	1.65	0.79	0.43	1.57
16	3.53	0.984	0.21	0.36	1.63	0.76	0.45	1.55
17	3.59	0.985	0.20	0.38	1.62	0.74	0.47	1.53
18	3.64	0.985	0.19	0.39	1.61	0.72	0.48	1.52
19	3.69	0.986	0.19	0.40	1.60	0.69	0.50	1.50
20	3.74	0.987	0.18	0.42	1.59	0.68	0.51	1.49
21	3.78	0.988	0.17	0.42	1.58	0.66	0.52	1.48
22	3.82	0.988	0.17	0.43	1.57	0.65	0.53	1.47
23	3.86	0.989	0.16	0.44	1.56	0.63	0.55	1.46
24	3.90	0.989	0.16	0.45	1.55	0.62	0.56	1.45
25	3.93	0.990	0.15	0.46	1.54	0.61	0.57	1.44

7.3 SPC set up

This form is used for configuration of the SPC, Statistical Process Control function. It is opened using the **Statistics SPC - Set Up** menu item.



Specify **Subgroup size** and if you want to use **Range** or **Standard Deviation**.

In the **SPC and TDA** frame, set up values for normal SPC and Trend Deviation Alarms.

Do the same for the **Short Term Trend** frame. These values will be used for calculations of all variables you choose to calculate on. See chapter: Data Collection for a description of the parameters.

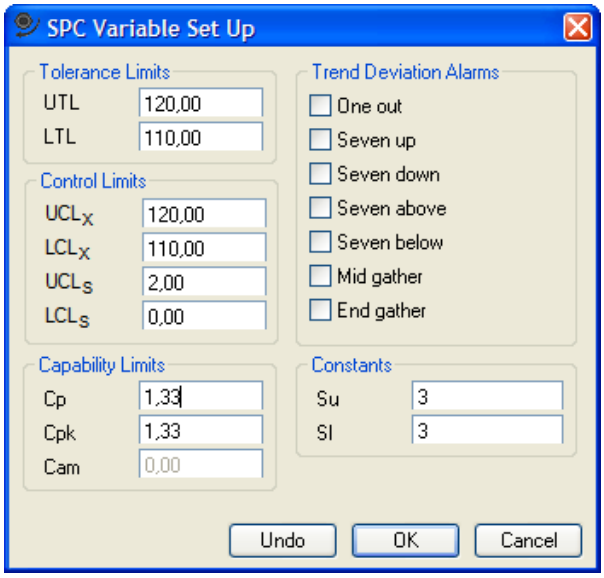
In the **Variables to collect** frame, add those variables you want to check. Select **Station**, **Bolt** and/or **Spindle**, **Program**, **Variable** and **Step**. For **Bolt** and/or **Spindle** you may specify **All**, this will add the variable for all bolts in the station. Press +. The variable shows up in the list box.

If more than one spindle can be used to tighten a bolt (defined in the Mode Table) you must define which spindle for the bolt to collect data for.

A PowerMACS system may contain up to 200 SPC variables and max 20 per TC/spindle.

To remove a variable from the list, select it and press the remove button ✕.

If you press the Information button i a dialog box is presented where you can enter various check values for the selected variable.



If you have marked **Auto recalculation of limits** on the first page, the **UCL_X**, **LCL_X, $UCL_{R/S}$** and **$LCL_{R/S}$,** fields will show current value of the control limits. If you disable automatic recalculation you can manually enter control limits in these fields.

Cam limit is only applicable if cam can be calculated.

Check one or more of the check boxes in the **Trend Deviation Alarms** frame to have the SPC and TDA data monitored for any of the following deviations:

- **One out**: One point outside control levels
- **Seven up**: Seven points consecutively increasing
- **Seven down**: Seven points consecutively decreasing
- **Seven above**: Seven points consecutively above average
- **Seven below**: Seven points consecutively below average
- **Mid gather**: More than 90% of the points in mid third
- **End gather**: Less than 40% of the points in mid third

If a deviation is detected an event of type SPC is generated and stored in the Event Log (see also View Event Log).

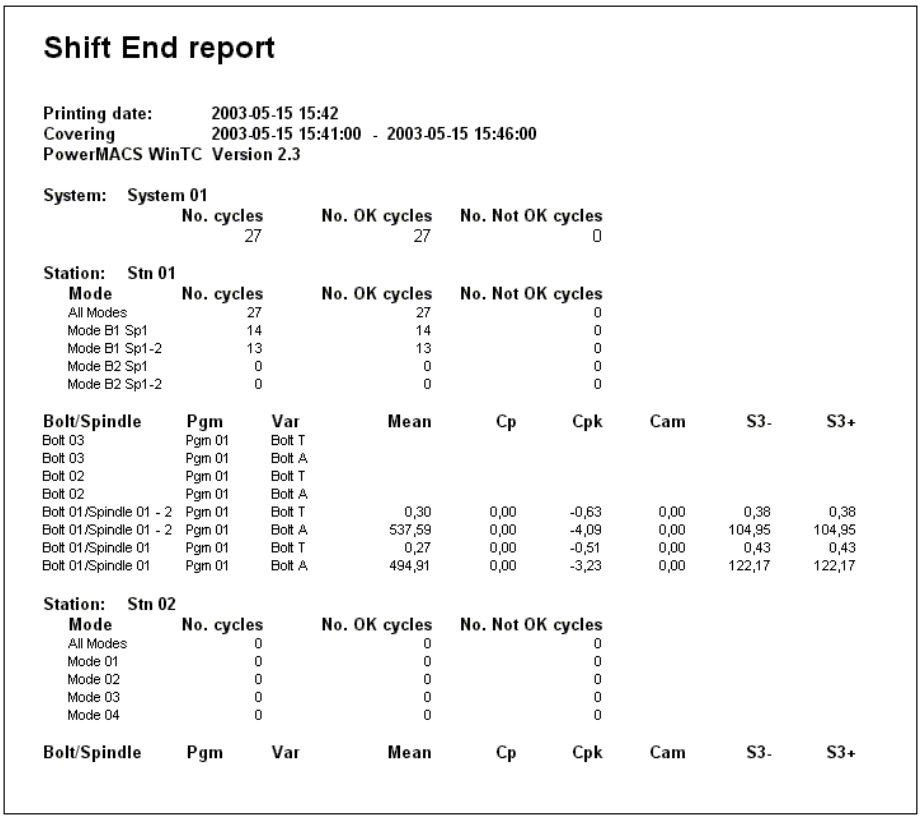
7.4 Shift Reports and Shift Set Up

It is possible to define automatic end of shift reports summarizing deviations and exceptions during the last shift. To use this function you must first define the shifts. This is done using the Shift Set Up form.

The report is produced automatically after each shift but can also be printed ”so far” during the current shift. The report can be previewed on the ToolsTalk PowerMACS screen or sent to a printer connected to it. Preview and print are invoked from the **Statistics** menu.

The report contains the following information per station:

- Total No. of cycles executed, No. of OK and No. of NOK.
- Total No. of cycles executed for each mode, No. of OK and No. of NOK.
- For each combination of program and bolt the following SPC data are listed for the bolt level result variables "Bolt T" and "Bolt A": Mean value, +3*Sdev, -3*Sdev, Cp, Cpk and Cam. To be included in the report the variables must first have been included using the SPC set up form.



Shift End report

Printing date: 2003-05-15 15:42
Covering 2003-05-15 15:41:00 - 2003-05-15 15:46:00
PowerMACS WinTC Version 2.3

System: System 01

No. cycles	No. OK cycles	No. Not OK cycles
27	27	0

Station: Stn 01

Mode	No. cycles	No. OK cycles	No. Not OK cycles
All Modes	27	27	0
Mode B1 Sp1	14	14	0
Mode B1 Sp1-2	13	13	0
Mode B2 Sp1	0	0	0
Mode B2 Sp1-2	0	0	0

Bolt/Spindle	Pgm	Var	Mean	Cp	Cpk	Cam	S3-	S3+
Bolt 03	Pgm 01	Bolt T						
Bolt 03	Pgm 01	Bolt A						
Bolt 02	Pgm 01	Bolt T						
Bolt 02	Pgm 01	Bolt A						
Bolt 01/Spindle 01 - 2	Pgm 01	Bolt T	0,30	0,00	-0,63	0,00	0,38	0,38
Bolt 01/Spindle 01 - 2	Pgm 01	Bolt A	537,59	0,00	-4,09	0,00	104,95	104,95
Bolt 01/Spindle 01	Pgm 01	Bolt T	0,27	0,00	-0,51	0,00	0,43	0,43
Bolt 01/Spindle 01	Pgm 01	Bolt A	494,91	0,00	-3,23	0,00	122,17	122,17

Station: Stn 02

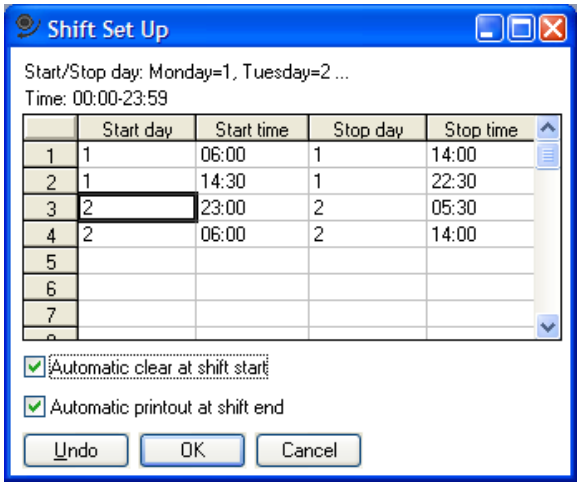
Mode	No. cycles	No. OK cycles	No. Not OK cycles
All Modes	0	0	0
Mode 01	0	0	0
Mode 02	0	0	0
Mode 03	0	0	0
Mode 04	0	0	0

Bolt/Spindle	Pgm	Var	Mean	Cp	Cpk	Cam	S3-	S3+

Some statistics can be automatically reported on a shift base.

7.4.1 Shift Set Up

The Shift Set Up form is used for to define shifts. It is invoked using the **Statistics - Shifts Set Up…** menu item.



Up to 21 shift periods can be defined. Each row in the above form represents a shift. Enter the day and time for when the shift starts and when it stops.

The days are expressed as numbers where 1 corresponds to Monday and 7 to Sunday. A shift may have different start and stop days.

The start and stop times must be entered using 24 h notation, i.e. as 00:00 to 23:59.

Check **Automatic clear at shift start** to have all shift statistics variables erased when a new shift starts. Clearing the shift statistics will reset the following:

- All counters (No. of cycles, No. of OK cycles, No. of NOK cycles) for each combination of station and mode used for the Shift Report.
- The Station level result variables "Total", "Total OK", and "Total NOK"
- The Bolt level result variables "Total", "Total OK", "Total NOK", "Total Type", "Total Type OK", and "Total Type NOK"

Check **Automatic printout at shift end** to automatically have the shift report printed when the shift ends.

See chapter: Shift Reports and Shift Set Up for an example on a typical shift report.

Note! In order to have the information on number of cycles executed in the report (OK as well as NOK) the following Station level result variables must be included in at least one reporter in the system (see chapter: Edit reporter for how to configure a reporter).

- Total
- Total OK
- Total NOK

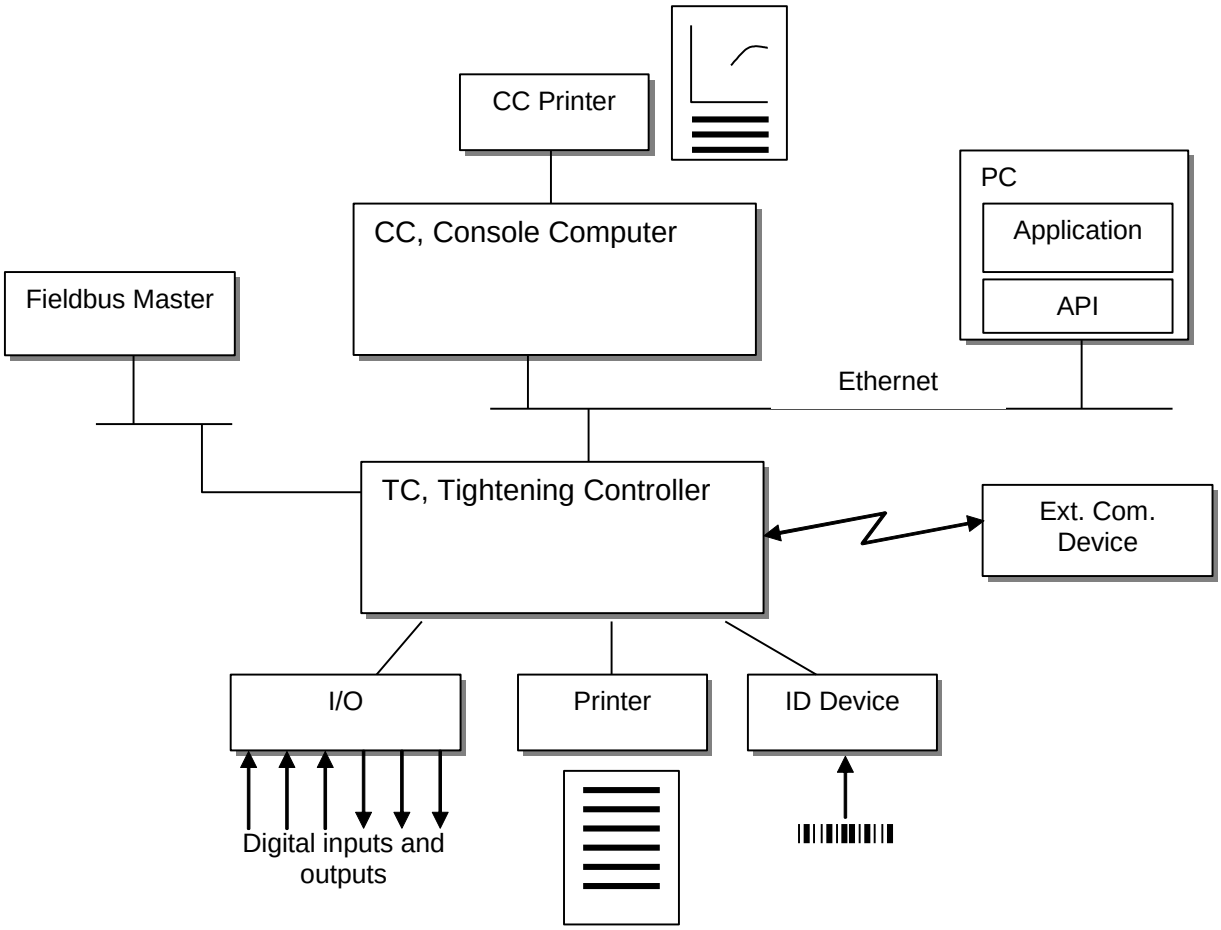
To have the SPC data included in the shift report the following Bolt level result variables must be included in at least one reporter in the system:

- Bolt T
- Bolt A

8 Peripheral Devices

8.1 Peripheral Devices - Overview

Peripheral devices are used to get information in and out of a PowerMACS system. The devices can be both of simple type, like printers, and of more complex type like a fieldbus, a serial communication protocol and the PowerMACS API interface.



Peripheral Devices

PowerMACS can handle the following devices:

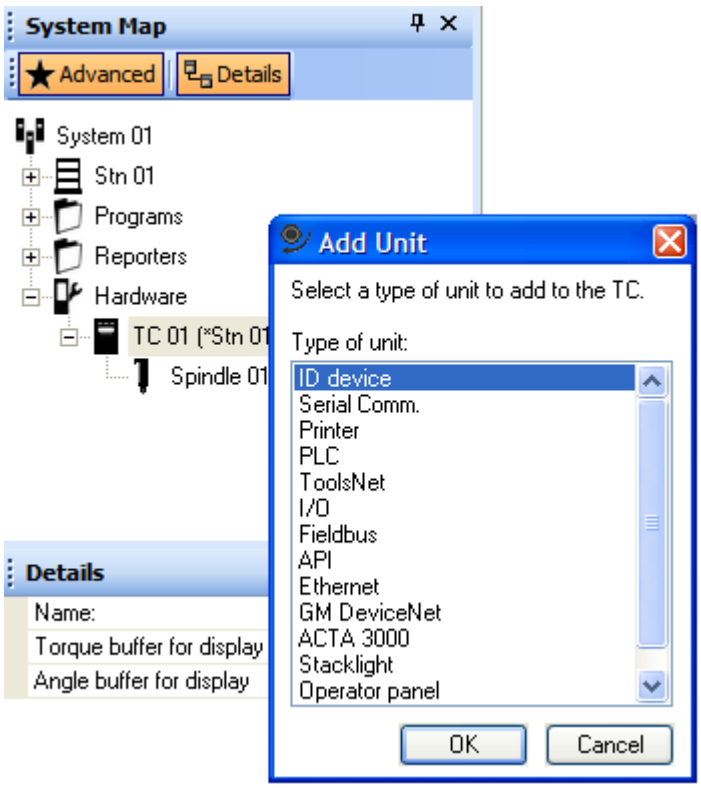
Device	Usage	Signal Interface
Printer or File on CC	Printout of cycle data, events, traces, SPC, etc.	Printer port on CC.
File on CC	Storage of cycle data end events in file.	File system on CC.
Printer on TC	Printout of cycle data and events.	Serial port on TC.
Stacklight (part of Standard Accessory Device)	Show status information.	Serial port on TC.
Indicator Box (part of Standard Accessory Device)	Show status information and control a station.	Serial port on TC.
Operator Panel (part of Standard Accessory Device)	Show status information and control a station.	Serial port on TC.
I/O Device	Binary data to/from the PLC.	Internal fieldbus port on TC.
ID device	Input for identification data and/or output of cycle data and events.	Serial port on TC.
Serial Communication Protocol Information	Data to/from PLC, output of cycle data, events and traces, read/write Setups.	Serial port on TC.
Fieldbus Interface	Data to/from PLC, output of cycle data, events and traces, read/write Setups.	Fieldbus interface board.
ToolsNet	Reporting cycle data and traces to Atlas Copco's ToolsNet server.	Ethernet, TCP/IP.
PLC	Reporting cycle data to the PowerMACS PLC (see chapter: PLC).	Internal
API, Application Programmers Interface	Data to/from PLC, output of cycle data, events and traces, read/write Setups, read/write Setup items.	Software package on a PC computer.
Ethernet Protocols	Receiving control and sending result from/to specialized servers like Open Protocol, DaimlerChrysler PFCS, FSH, DC PLUS etc.	Ethernet, TCP/IP.
GM DeviceNet	Receiving control and sending result from/to a an overriding controller according to GM's specification "GAE Electric Nutrunner Controller Design Criteria"	DeviceNet fieldbus and/or local I/O via the internal fieldbus port on TC.
ACTA 3000	Calibration of spindle torque transducer.	Serial port on TC.

The following chapters will describe how these devices are added and used.

8.2 Add a device

You can have a total of 21 devices in a PowerMACS system and up to 6 can be connected to each TC.

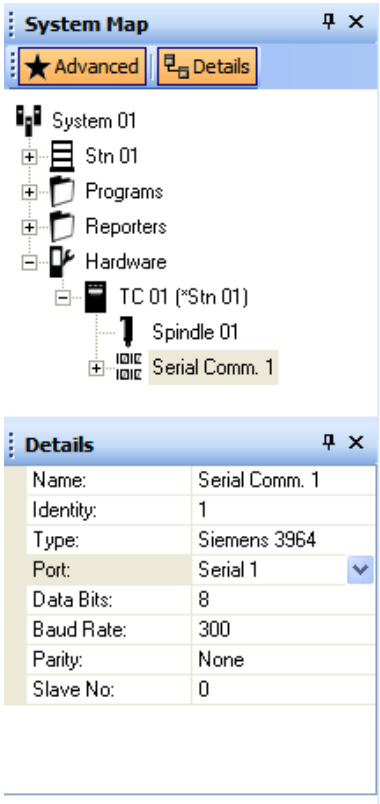
To add a device, select a TC in the System Map, right click and select **Add Device** from the popup menu. This will bring up a list of devices that can be added



You will be asked to give the new device a name. This name will later be referred to when you create a Reporter, which will be used to select and format the data sent to the device.

When you have added a device you should configure it. Which parameters that is available depends on the type of device.

Example for an ID device:



For it you must specify Type, which is one of Barcode, Escort P&F and Escort AB.

Select a suitable port. There are several ports on a Tightening Controller that can be used for in and output of data.

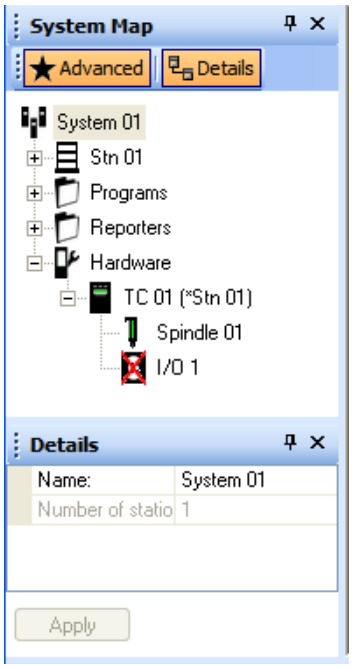
Port	Usage	Signal Interface
Serial 1	General serial communication	RS-232, RS-485
Serial 2	General serial communication	RS-232, RS-422, ST-Bus
CAN I/O port	I/O Device communication	CAN with DeviceNet

For the selected port you must also specify its characteristics: Baud Rate (300-115 200), Parity (None, Even or Odd) and Data Bits (7 or 8). Number of Stop Bits is always 1.

Note! Due to technical reasons you can only use two different baud rates on each TC.

8.3 Status of a device

During performing its task each device checks for errors or malfunction. The result is shown in the **System Map**.



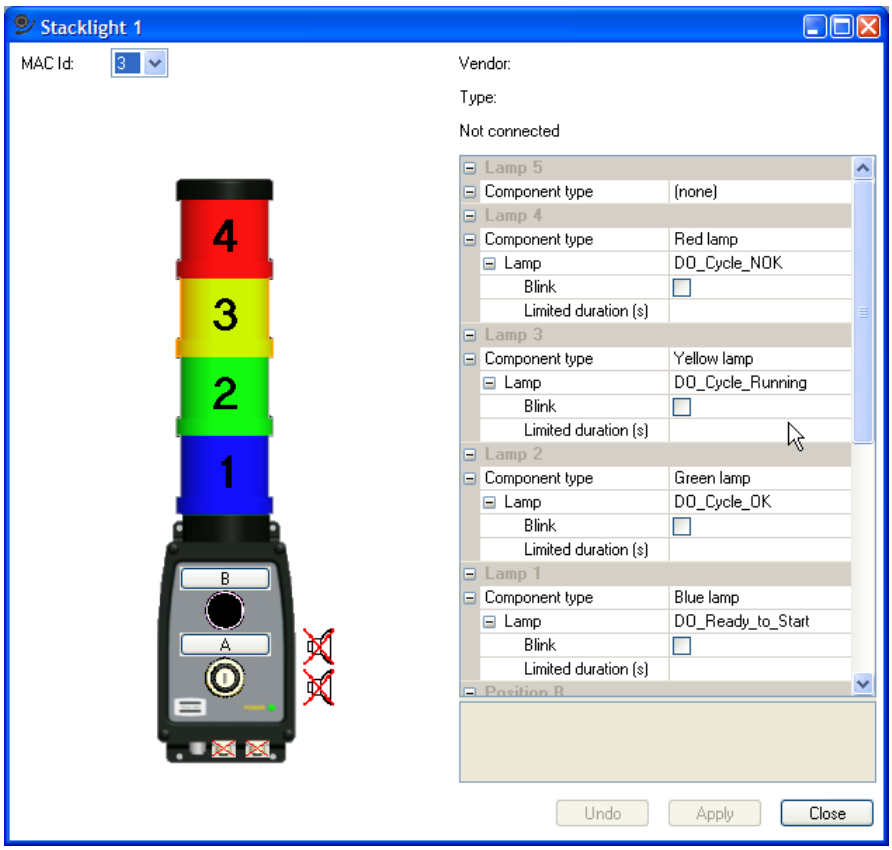
Devices not working OK are shown with a red cross.

8.4 Standard Accessory Devices

The standard accessory devices concists of Stacklight, Indicator Box and the Operator Panel. They are all communicating using the DeviceNet protocol and can essentially be considered to be specialized I/O devices.

8.4.1 Stacklight

To access the stacklight configuration window, add a stacklight device and double-click on it in the system map.



Select the correct **MAC Id** for the Stacklight. When a new Stacklight device is added a simple default configuration is provided.

Configure the Stacklight by selecting the component you wish to edit (lamp, buzzer or button) in the property grid on the right side or by clicking on the actual component in the Stacklight image (this will cause the correct property to be selected).

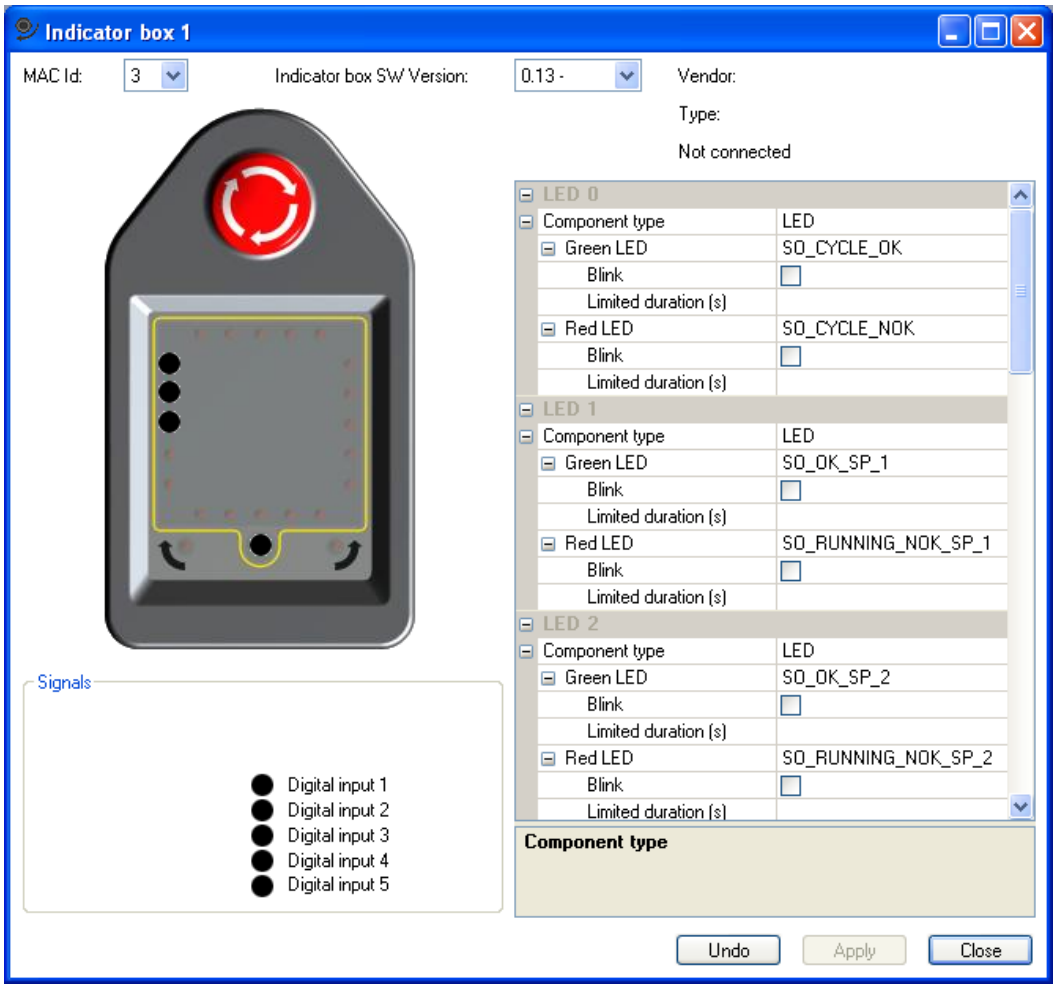
Once a component is selected configure the digital IO signal to connect to the component and additional properties (flashing, duration etc.).

It is possible to select the number of lamps on the indicator box by selecting the “None” value as **Component type**.

When connected to a system the form will show show the signal values in realtime and no editing is possible.

8.4.2 Indicator Box

To access the indicator box configuration window, add a indicator box device and double-click on it in the system map.



Select the correct **MAC Id** for the Indicator Box. Select the correct **Indicator box software version**. Software versions up to 0.12 have a configuration in the Indicator box which specifies ten lamps to be used. Later software versions allow the user do decide exactly which of the lamps should have a specific function.

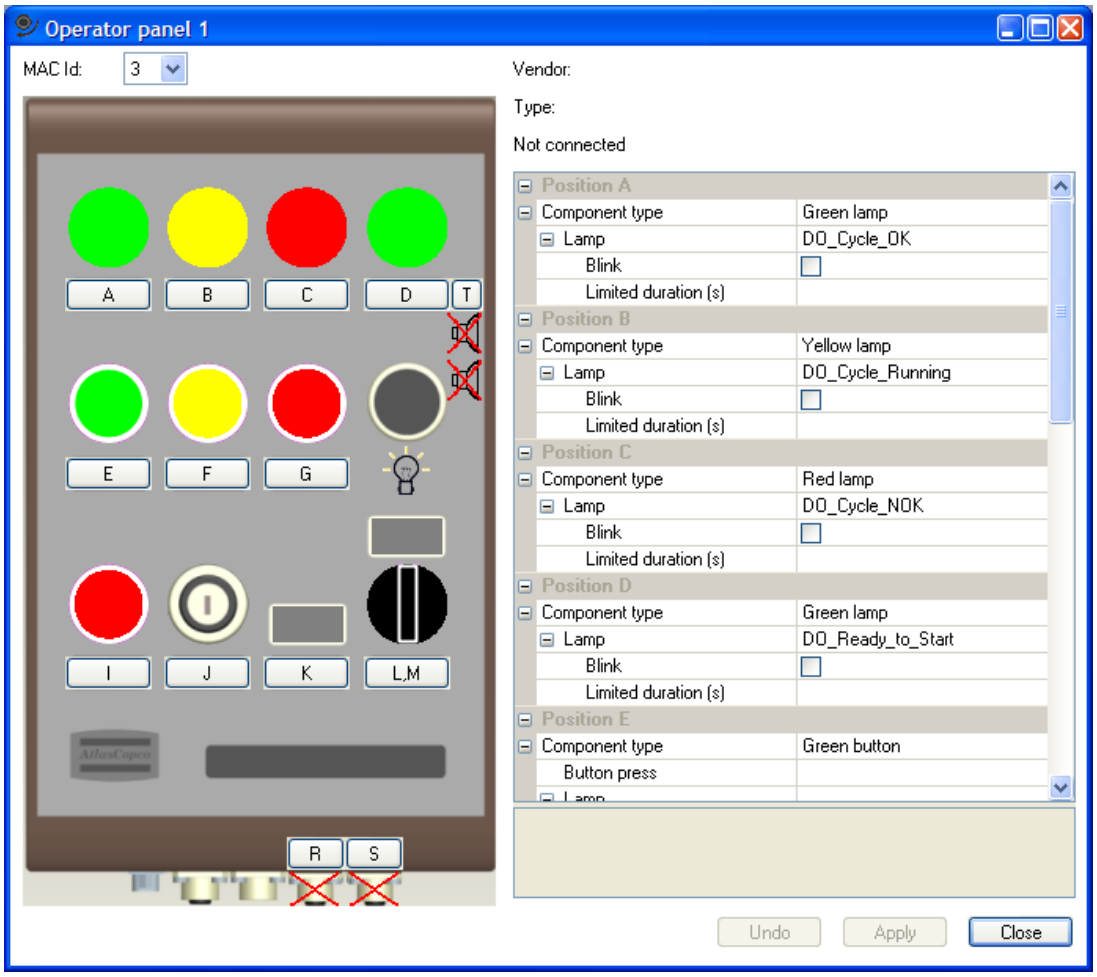
Configure the Indicator Box by selecting the component you wish to edit (lamp or digital input) in the property grid on the right side or by clicking on the component in the indicator box image or LED signal list.

Once a component is selected configure the digital IO signal to connect to the component and additional properties (flashing, duration etc.).

When connected to a system the form will show show the signal values in realtime.

8.4.3 Operator Panel

To access the operator panel configuration window, add an operator panel device and double-click on it in the system map.



Select the correct **MAC Id** for the Operator Panel. When a new Operator Panel device is added a simple default configuration is provided.

Configure the Operator Panel by selecting the component you wish to edit (lamp, buzzer or button) in the property grid on the right side or by clicking on the component in the operator panel image.

Once a component is selected configure the digital IO signal to connect to the component and additional properties (flashing, duration etc.).

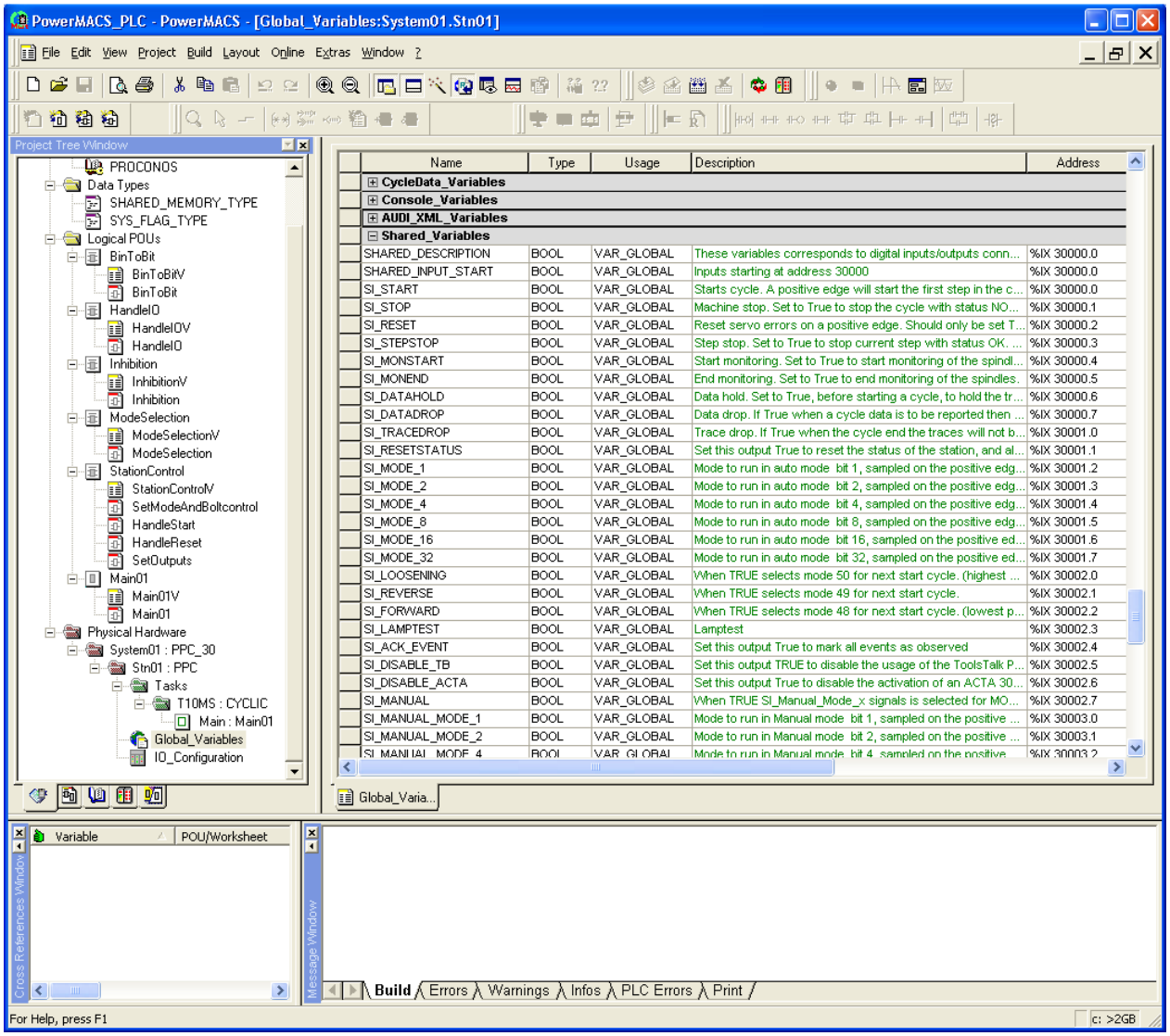
A special integer **Mode** signal is provided to use with LED displays to show the actual mode number selected. Use the **DI_Manual** signal to override other devices mode selection.

When connected to a system the form will show show the signal values in realtime.

8.5 I/O Device

An I/O device is used to communicate with hardware modules for digital input and output signals. For description on I/O devices and how to set up, see chapter: I/O.

The I/O device, as well as Stacklight, Indicator Box, and Operator Panel, can map all boolean signals located in the Digital_In_Out group and in the Shared_Variables group. Boolean outsignals starting with “SO_” in the Fieldbus_Variables group can also be mapped by these devices since they belong to the shared signals. Se PLC Areas for more information about these areas in the PLC.



8.6 Printer on TC

A Printer device connected to a TC can be used to print cycle data and events via a serial port.

Most printers will work as long as they are made for printout of normal ASCII characters. There is no handshake protocol or modem signaling needed.

Add a Printer device as described in previous chapter. Set up which port to use and the port characteristics.

Now you have added a printer device. To get data on it you must also create a **Printer** Reporter. How to do this, and connect it to your Printer device, is described in the New reporter chapter. How to make the Reporter format the result as you wish is described in chapter: Edit reporter

8.7 Printer or File on CC

A Printer device connected to a CC can be used to print cycle data and events, but also for printout of graphical traces and SPC charts.

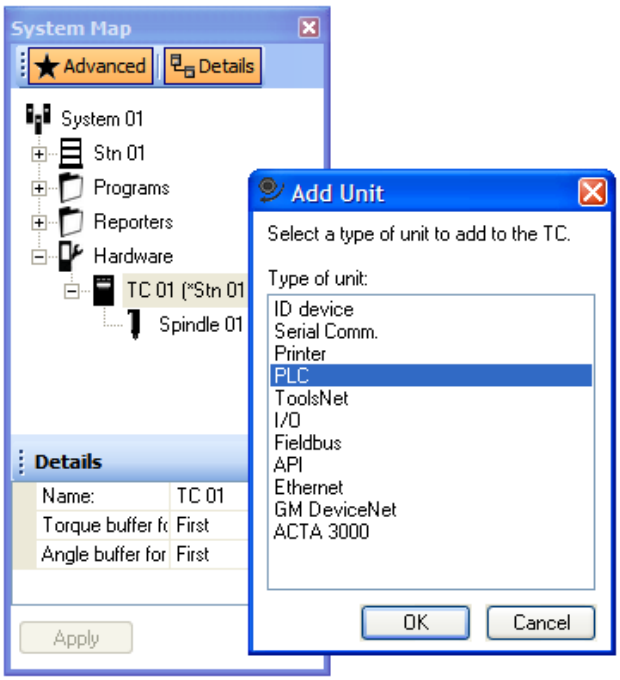
To get data to the printer it you must create a **ToolsTalk Printer** Reporter. How to do this is described in the New reporter chapter. How to make the Reporter format the result as you wish is described in chapter: Edit reporter

Note! The ToolsTalk PowerMACS program must be running if order to be able to print on a printer connected to the CC.

8.8 PLC

Using the PLC device it is possible to read cycle data from inside the PowerMACS PLC (see chapter: PLC).

To enable this function you must first add a PLC device, as described in chapter: Add a device, to TC that runs the PLC, i.e. the first TC in a station. Note that the PLC device is only a logical device, it requires no additional hardware to be connected to the system.



Secondly you must create New reporter and connect it to the PLC device.

Use the reporter to select what result variables to include in the cycle data, see Edit reporter for how to do that.

Finally you must declare variables within the PLC that corresponds to the layout of the Reporter.

Configuring a reporter with Type of layout set to "Standard", **Additional new lines** to 0 (or empty), **Byte order** and **Status format** to "Normal", **Float format** to IEEE754, and the following result variables:

Station variables:

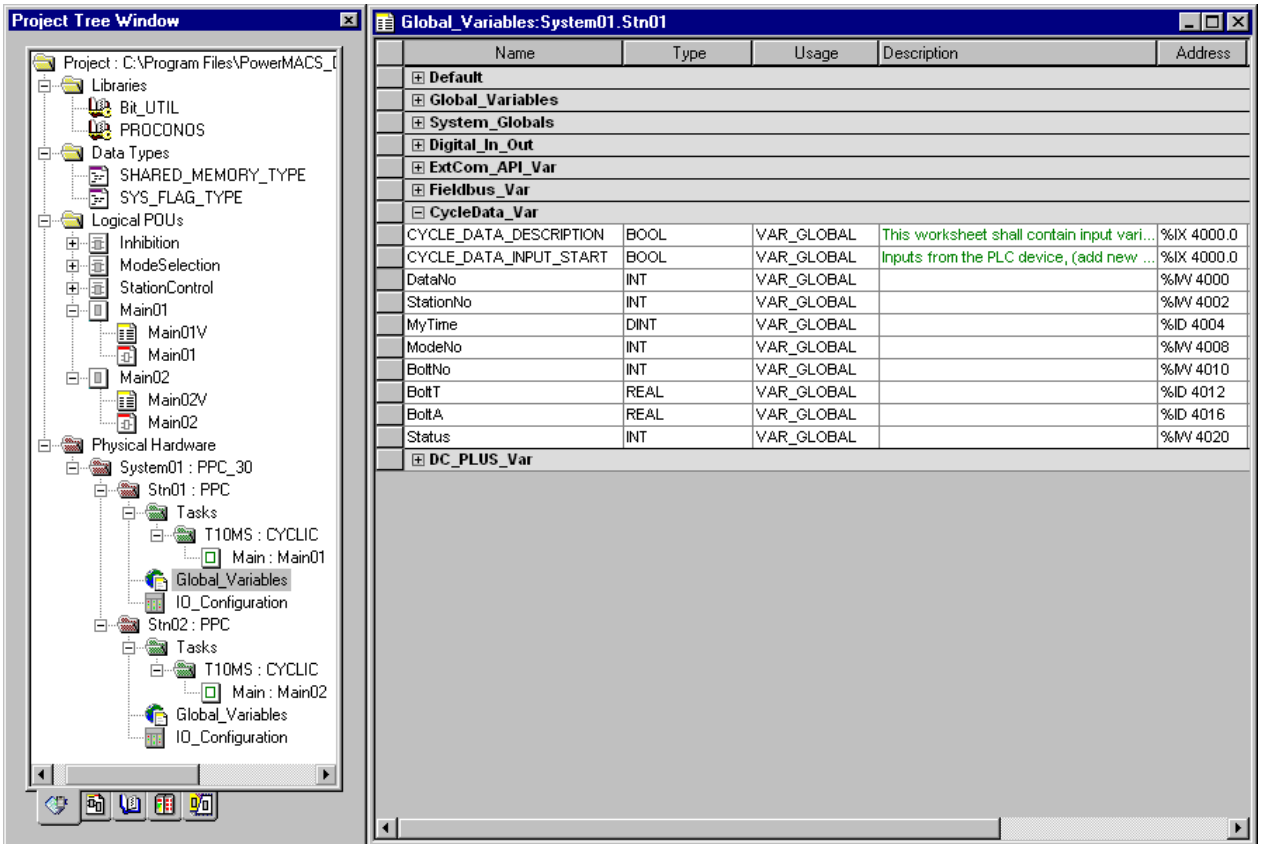
Variable	Order	Width	Dec	Type	Text	Lines
Data No	1			I2		
Station No	2			I2		
Time	3			I4		
Mode No	4			I2		

Bolt variables:

Variable	Order	Width	Dec	Type	Text	Lines
Bolt No	1			I2		
Bolt T	2			F		
Bolt A	3			F		
Status	4			I2		

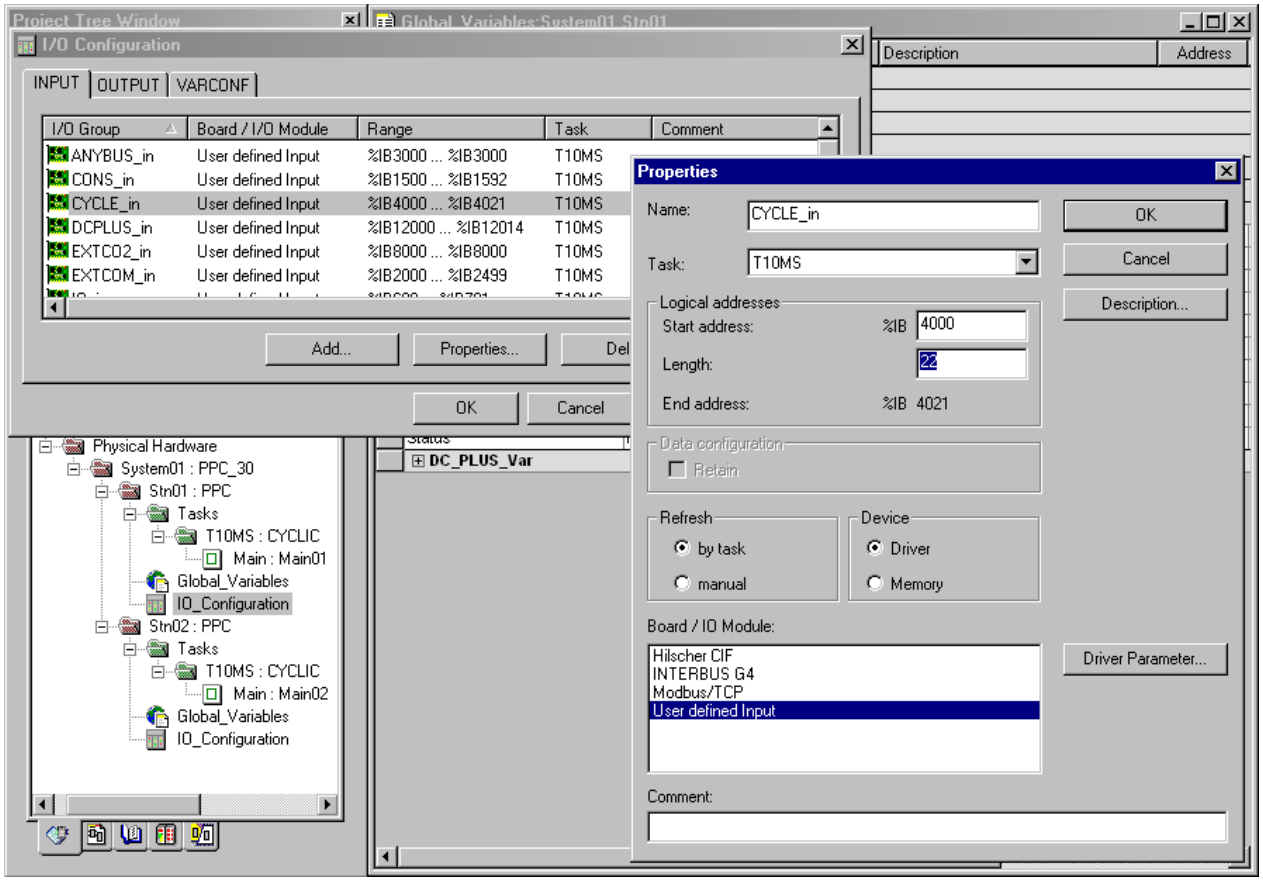
You must also enable and configure the drivers used for accessing the PLC data. This is done using the **IO_Configuration** worksheet for station in question.

The above setup (for a station containing one bolt) corresponds to the declaration of the following PLC variables in the **CycleData_Var** section of the Global_Variables worksheet for the station in question:



You must also configure the PLC drivers used for accessing the PLC data. This is done using the I/O Configuration dialogue which is invoked by double-clicking on the icon for **IO_Configuration** worksheet of station in question in the Project Tree.

In the /O Configuration dialogue select the I/O Group CYCLE_in located on the tab marked INPUT and press the **Properties** button.



Change the value of **Logical addresses – Length** so that it corresponds to the size of your cycle data, in this case 22.

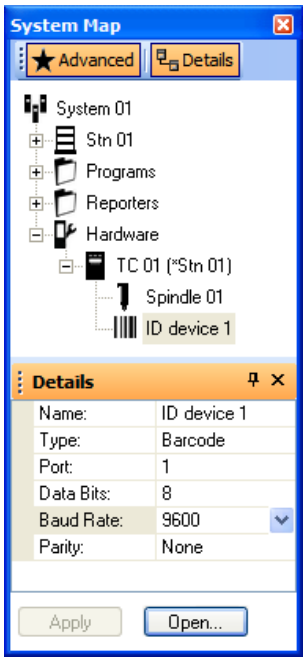
The Start and End addresses shown (here 4000 and 4021) are logical, meaning that a PLC variable later mapped to the logical address 4000 will correspond to the first byte of the cycle data.

8.9 ID device

ID devices come in two forms: input only and both input and output. An input device could be used to get identification of a work piece. The read data could include information on how the tightening should be performed. The identification could also be used to identify the result data.

An ID device, which can be written, can be used to store crucial data (status, final torque etc.) along with the work piece. This can then later be read in, for example, a repair station.

See chapter: Add a device for how to add an ID device.



Use Type to specify the type of ID device to use. Currently the following ID device types are supported:

- **Barcode** - Barcode scanner, any brand that uses a serial input.
- **Escort P+F** - Escort memory of type Pepperl+Fuchs, MVI-D2-2HRX
- **Escort AB** - Escort memory of type Allen Bradley, Intelligent Antenna 2750-ASP* together with code tag 2750-TAU40
- **Omron** - Escort memory of type Omron V600 RFID system (V600-CA1A-V and V600-CA2A-V)
- **Euchner** - Euchner Electronic-Key-System reader of type EKS-A-ASX

A Barcode scanner is an input only device. It can be used to read bar codes attached to a work piece. An Escort Memory is an ID device that can be used both for input and output.

All ID device types are connected to the PowerMACS TC using a serial communication interface. Use Port to specify of PowerMACS communication ports the device is connected to and **Baud Rate**, **Parity** and **Data Bits** to configure its characteristics.

Press the **Advanced…** button to open the ID device Set Up form from where you can set all type specific parameters, like at which addresses to read/write from/to for an escort memory device. See Type specific parameters below for a description of these.

To make use of the data read from an ID device it must be transferred to a Station. This connection is configured using the Advanced Station Settings form, see chapters: Work piece identifier and Multiple identifiers. Note that it is possible for several Stations to read from the same ID device.

The data read from an ID device is also available to the PowerMACS PLC making it possible to process it as wished.

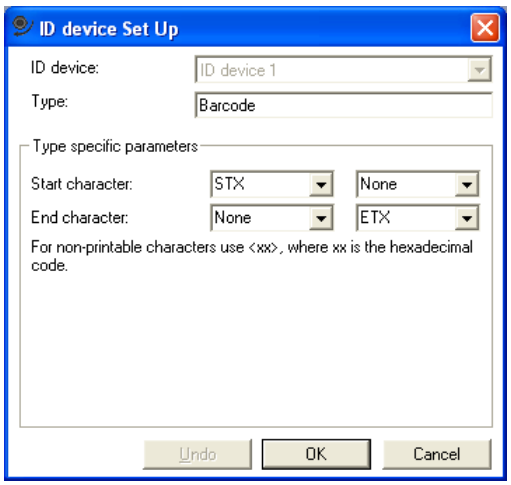
When the device is connected to a Station the data read from it is also available to the Stations PLC for processing. The PLC can also be used to control and supervise reading of the data. The PLC functions available are described in chapter: ID device variables and Multiple identifier variables.

If result data should be written to the ID device (an escort memory) you must set up how the data should be formatted. This is done by connecting a Reporter to the device. See chapter: New reporter for how to create a new Reporter and/or chapter: Edit reporter for how to configure it. Remember to specify if data should be written as readable characters or in binary format. The writing of the escort memory is done as soon as the cycle is ready.

8.9.1 Type specific parameters

Type specific parameters are opened by clicking the **Open...** button located on the **Details** form.

8.9.1.1 Barcode scanner

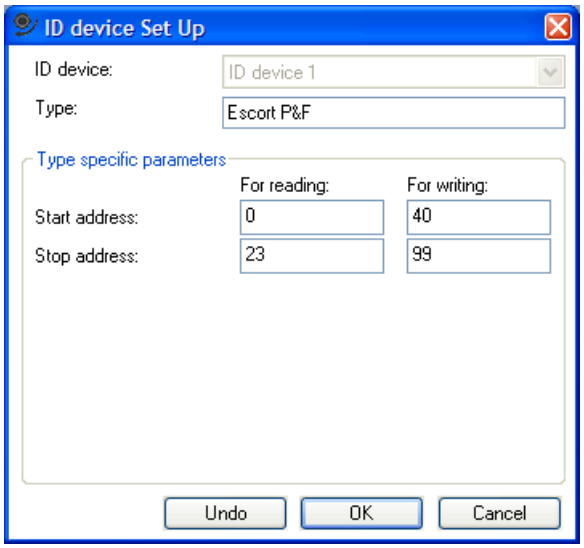


An ID device of type **Barcode** normally handles a barcode scanner. Since PowerMACS allows you to freely choose which characters that are used to frame the scanned ID string it supports most types of scanners.

The character entered as **Start character**, if any, must be the first character returned from the device. If not the device is marked erroneous until a correct string is scanned.

Enter as **End character** the character that marks the end of returned string. Non-printable characters are entered using their hexadecimal code enclosed between a pair of "<" and ">", for example "<09>" for a horizontal tab.

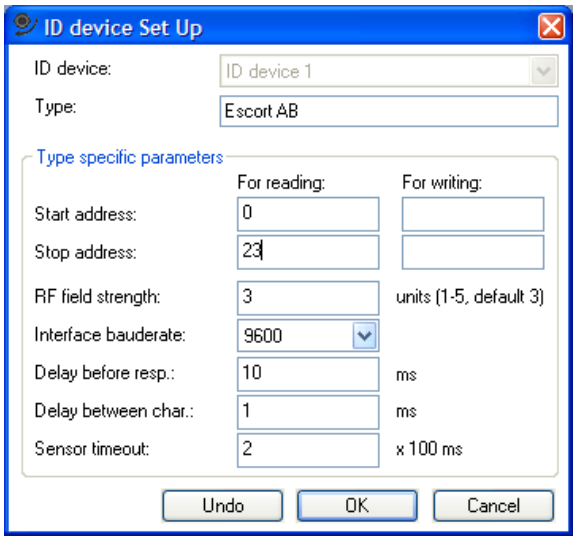
8.9.1.2 Pepperl + Fuchs escort memory



For an ID device of type **Escort P&F**, that is, a Pepperl+Fuchs escort memory, you must specify where in the data tag to read the ID string from and where to write cycle data to.

Use the two pairs of **Start address** and **Stop address** to do this. The first address in the tag has address zero (0). For reading you can specify up to 40 cells and for writing up to 500.

8.9.1.3 Allen Bradley escort memory



The ID device type **Escort AB** corresponds to Allen Bradley's Intelligent Antenna 2750-ASP* together with code tag 2750-TAU40.

For the Allen Bradley Intelligent Antenna escort memory, you must specify where in the data tag to read the ID string from and where to write cycle data to. Use the two pairs of **Start address** and **Stop address** to do this. The first address in the tag has address zero (0)

RF field strength controls the power of the signal transmitted by the antenna and may need to be adjusted depending on the distance between the antenna and data tags.

The parameters **Delay before resp**, **Delay between char** and **Sensor timeout** should normally be kept at their default values.

For more information on these parameters see Allen Bradley's User Manual for the Intelligent Antenna.

8.9.1.4 Omron escort memory

The ID device type **Omron** corresponds to Omron V600 RFID system. Currently the two models V600-CA1A-V (RS-232C interface) and V600-CA2A-V (RS-422 interface) are supported.

Note! V600-CA2A-V (RS-422) can only be connected to port X104 since it is the only port that supports RS-422. For other ports an external converter must be used.

For an **Omron** ID device you must specify where in the data tag to read the ID string from and where to write cycle data. See Reading an ID string and Writing result data below for how to do this.

Parameter **Antenna** controls which of the two antennas supported by the V600 serial interface controller to use. Currently only 1 (one) is supported.

Reading an ID string:

For Omron devices the ID String may consist of two different parts, the *main part* and an *optional number part*.

Parameters **Start address** and **Stop address** defines the address area on the tag where the *main part* is located. The *main part* must be an ASCII string. The first address in the tag has address zero (0) and up to 40 bytes can be read.

Parameters **Start address 2** and **Stop address 2** defines the address area on the tag where the *optional number part* is located. The parameter **ID string conversion** controls how this data is converted to a numerical value and it takes one of the following values:

- **Mitsubishi**: The *optional number part* is interpreted as a four byte integer having a special Mitsubishi BCD format where the decimal value 1234 is formatted as four consecutive bytes having the values 0x04, 0x03, 0x02, 0x01 with the first byte (0x04) is located at lowest address.

- **ASCII**: The *optional number part* is assumed to be a four character string consisting of decimal digits only. That is, the decimal value 1234 is formatted as four consecutive bytes having the values 0x31, 0x32, 0x33, 0x34 with the first byte (0x31) is located at lowest address.

The value read from the tag is converted to a four character decimal number (example: value 47 gives the string "0047"). Leaving either **Start address 2** or **Stop address 2** blank disables reading of the *optional number part*.

If the *optional number part* is used it is concatenated to end of the *main part* before the resulting ID string is sent to the station and the PowerMACS PLC.

Example: If the *main part* is the string "ABCDE" and decimal value 47 is read as *optional number part* the resulting ID String is "ABCDE0047". This string is sent to the station and PLC as a response to an IDREAD (PLC output) request.

Writing result data:

Start address and **Stop address** defines the address area of the tag where result data is written if a reporter is assigned to the device. the offsets are relative to **Start address**.

Limitations:

A V600 controller cannot be shared between different ID devices. Every TC that should write to a tag must have a separate V600 controller.

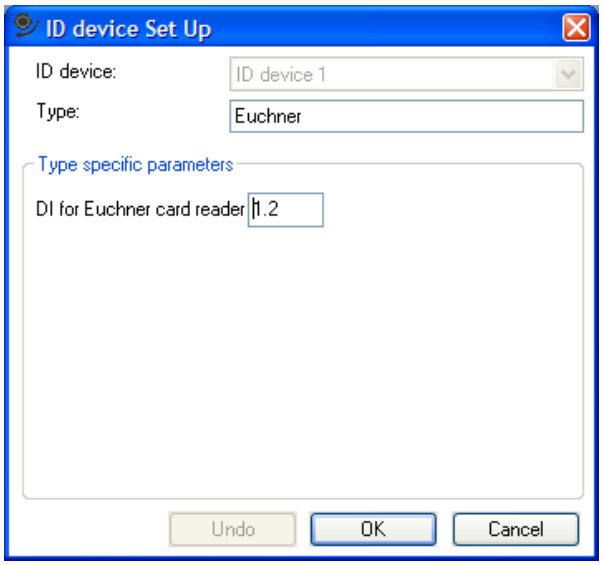
Only one of the V600 controllers' antennas can be used. Which antenna to use is configurable.

The system configuration for 1 to N connections (available only for V600-CA2A) is not supported.

The command used for reading and writing data in the EM is Read (RD) and Write (WT), which means that there is no support for Auto Read (AR), Auto Write (AW), Polling Auto Read (PR) or Polling Auto Write (PW).

Since auto reading is not supported an external signal has to be used to inform the PLC that the tag is in place for reading or writing.

8.9.1.5 Euchner card reader



An ID device of type **Euchner**, that is, a Euchner Electronic-Key-System reader of type EKS-A-ASX, is equipped with a 24V output signal (data carrier active) that is used by PowerMACS to know when the reader has a card. When a card is inserted in the reader, the 24V signal goes high. This triggers PowerMACS to start communicating with the card reader and fetch the data from the card. Whenever the signal is set low PowerMACS clears the read data.

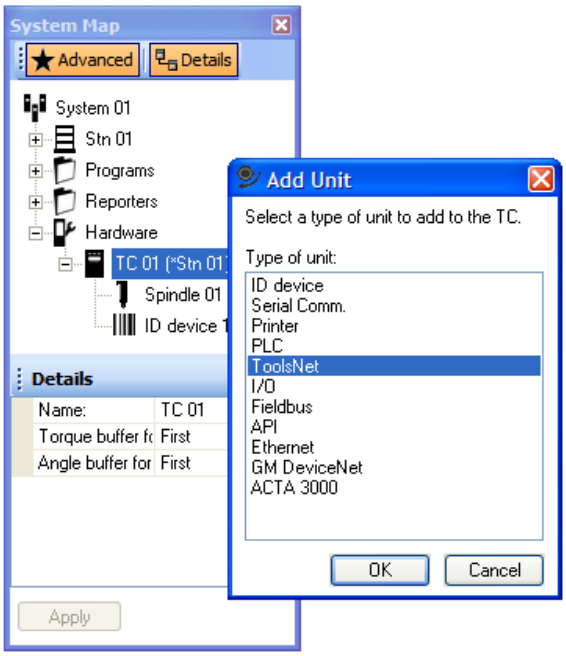
Use **DI for Euchner card reader** to specify which PowerMACS I/O input that the "data carrier active" signal is connected to.

Note! The Euchner device must be configured to use Baud Rate 9600, Parity = Odd and Data Bits = 8.

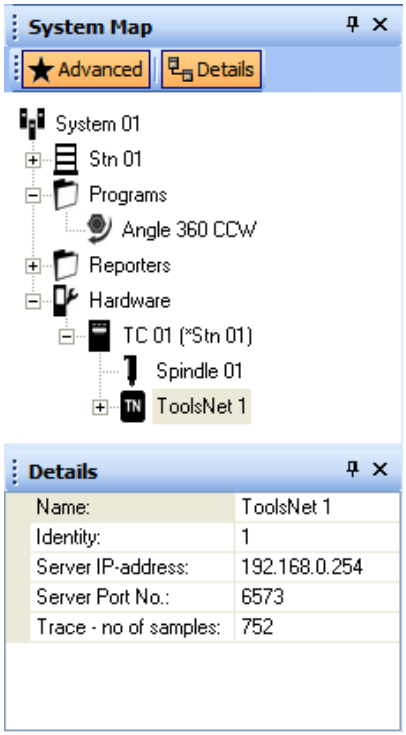
8.10 ToolsNet

Using a ToolsNet device you may report cycle data and/or traces to an Atlas Copco ToolsNet server over the Ethernet.

To enable this function you must first add a ToolsNet device, as described in chapter: Add a device.



Select the created device and adjust its parameters if necessary.



Use **Name** to specify the name of the device. This name should be referred to when you create a reporter for the new ToolsNet device using the New reporter form.

Server IP-address should be set to IP-address of the computer on which the ToolsNet server is installed.

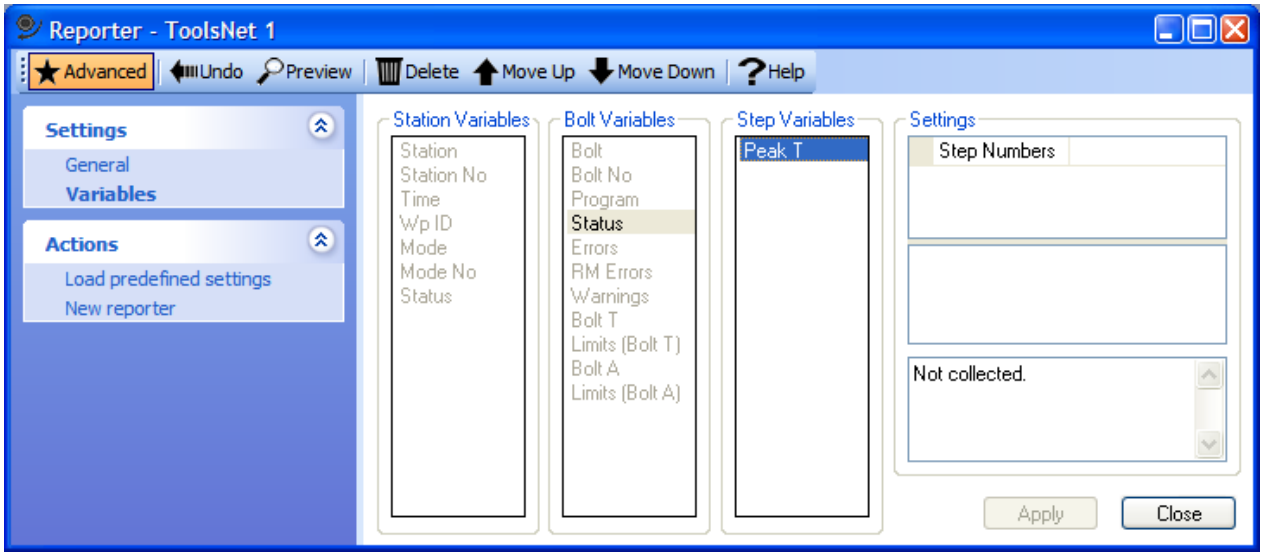
The value of **Server Port No** depends on the configuration of ToolsNet and should only be changed after consultation with your network manager. Default values for the port number are suggested by the system when the protocol version is changed.

Trace – no of samples controls how many samples of a PowerMACS trace that are sent to the Toolsnet server. It can be set to a value in the range [100…2000] and the default value is 752.

When ToolsNet is used together with DC PLUS it is possible to have the "PN" or "KN" prefix of the "Wp. Id" result variable removed when it is sent to the ToolsNet server. This filter is activated by checking the **Filter PN/KN** checkbox (displayed only if the system contains a DC Plus device).

The ToolsNet device does not require any additional hardware to be connected to the TC since it uses the TCs standard Ethernet connector. However, you must make sure that the network between your PowerMACS system and the ToolsNet server is correctly configured. See PowerMACS Ethernet Manual [3] for how to do this.

Create a new Reporter for the ToolsNet device and use it to set up what data to report to ToolsNet. The ToolsNet reporter is somewhat different from reporters used for other devices since you can only select what variables to include, not format them. Certain variables can not be removed from the reporter, they are marked as gray in the reporter window.

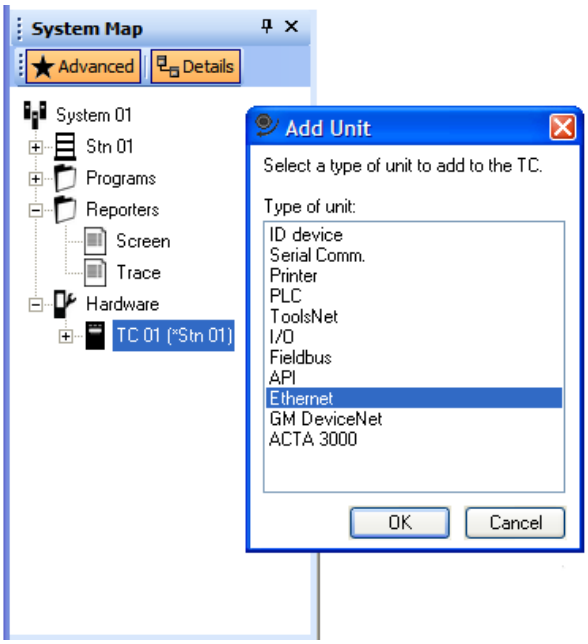

Reporter - ToolsNet 1
Advanced
Undo
Preview
Delete
Move Up
Move Down
Help
Settings
General
Variables
Actions
Load predefined settings
New reporter
Station Variables
Station
Station No
Time
Wp ID
Mode
Mode No
Status
Bolt Variables
Bolt
Bolt No
Program
Status
Errors
RM Errors
Warnings
Bolt T
Limits (Bolt T)
Bolt A
Limits (Bolt A)
Step Variables
Peak T
Settings
Step Numbers
Not collected.
Apply
Close

8.11 Ethernet Protocols

Today most factories have a local area network, LAN, for distribution of data. These networks are often based on Ethernet and with TCP/IP as link protocol. PowerMACS is capable of using such LAN for communication to and from a central server. What functions possible to use differs between protocols but may include things like:

- Reporting cycle data
- Control of tightening
- Reporting alarms

To enable this function you must first add an Ethernet protocol device, as described in chapter: Add a device.

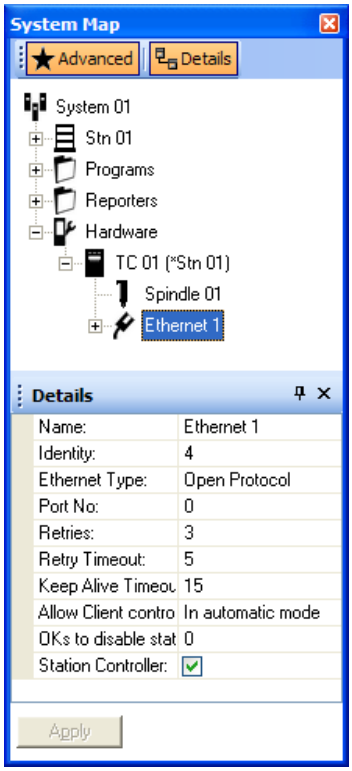


Add the device to a TC that has a PLC, i.e. the first TC in a station.

With the exception of DC PLUS and Audi XML it is possible to set up multiple Ethernet devices on any TC in the system. The Audi XML device is limited to one per system and must be on the System TC. For DC PLUS there can be one device per station and they must be located on the respective station TC. However, Ethernet devices of other types can coexist on the same TC as the Audi XML and DC PLUS devices.

Select the created device and adjust its parameters if necessary.

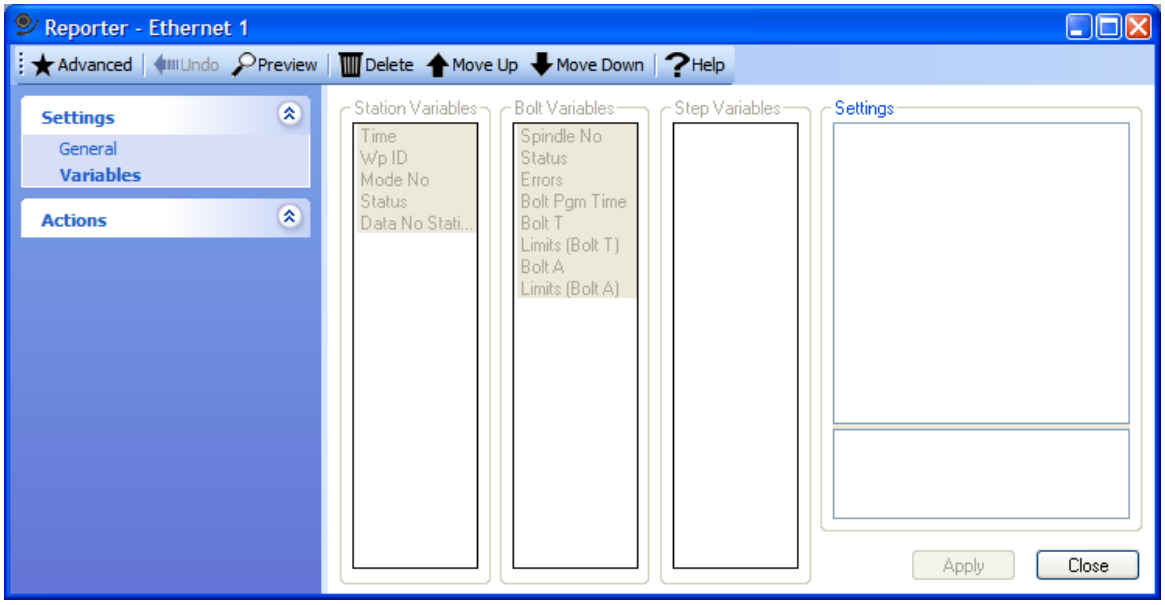
Example:



Use **Name** to specify the name of the device. This name should be referred to when you create a reporter for the new ToolsNet device using the New reporter form.

Set up which **Ethernet Type** of protocol to use, and which **IP-address** the server has.

A new Reporter is created automatically for the Ethernet protocol device. The reporter is different from other reporters in that sense that what variables to include are predefined and cannot be changed.



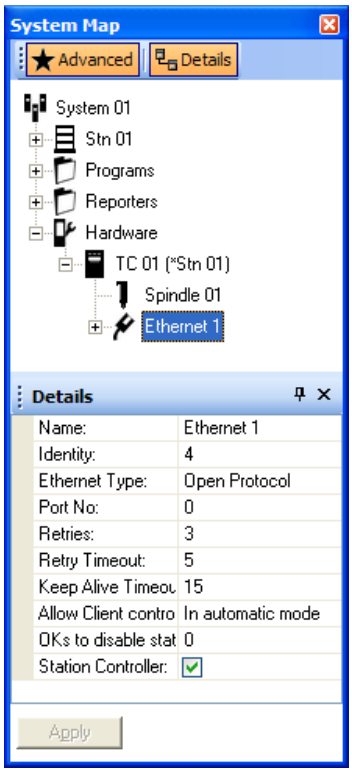
In chapters below is described how to configure the different Ethernet protocol devices. The protocols are standard protocols, which are defined by major car manufacturers. It is also described what functions are included and how they are mapped on PowerMACS functions.

The Ethernet protocol device does not require any additional hardware to be connected to the TC since it uses the TCs standard Ethernet connector. However, you must make sure that the network between your PowerMACS system and the Ethernet server is correctly configured. See PowerMACS Ethernet Manual [3] for how to do this.

8.11.1 Open Protocol

Note! This protocol has been renamed it was earler called FFCCP and supported revision 4.7 of FFCCP, the Ford Fastening Controller Common Protocol. The FFCCP functionality still exist in this protocol.

PowerMACS supports a subset of Open Protocol described in the document *9836 4415 01 Open Protocol* (se Supported message types below).



Port No specifies the port on which PowerMACS listens for incoming requests for Open Protocol.

Retries specifies how many retries should be done in case a telegram has not been able to send. **Retry Timeout** specifies the timeout between retries.

Keep Alive Timeout is the timeout used to detect the absence of a client. If no telegram has been received within the timeout period, the TC closes the connection.

OKs to disable station specifies the number of cycles with status OK or OKR that may be run while in Automatic mode before the station is automatically disabled. NOK cycles are not counted. If **OKs to disable station** is greater than zero then the station is disabled at power up and whenever a connection to Open Protocol is established. If left empty, or zero, then the station is only disabled if explicitly ordered so by Open Protocol. All tightening made during a batch will have the same VIN number, sampled at the

first start. The actual value of the OK, or batch, counter is reset when a "Select Parameter set" message (MID 0018) is received from Open Protocol but is unaffected by Enable/Disable commands.

Allow Client control specifies how control messages (see below for a listing which these are) from the Open Protocol client should be treated by the tightening controller. Choose one of the following alternatives:

- **Always**: Select this alternative if control messages should be accepted regardless if the station is in Automatic or Manual mode.
- **In Automatic mode**: (Default). Select this alternative if control messages only should be accepted when the station PLC output MODE is equal to zero.
- **Never**: Select this alternative if control messages never should be accepted.

See Automatic/manual mode for a description on the differences between Automatic and Manual mode.

The following messages from the Open Protocol client are regarded as control messages:

- MID 0018, "Select Parameter set"
- MID 0042, "Disable tool"
- MID 0043, "Enable tool "
- MID 0050, "Vehicle Id Number download request"

When these are received in a state when they are not wanted according to the value of the **Allow Client control** parameter PowerMACS will respond with a “Command Error" (MID 0004) message with the field "Error" set to 95 (Reject request, PowerMACS is in manual mode).

Station Controller specifies if this device is allowed to read/write user data to/from the PLC. If this is checked the following messages from the Open Protocol Client are allowed.

- MID 240, “User data download”
- MID 241, “User data subscribe”
- MID 242, “User data upload”
- MID 243, “User data upload acknowledge”
- MID 244, “User Data upload unsubscribe”

8.11.1.1 Automatic/manual mode

The station controlled by the Open Protocol device is always in either Manual or Automatic mode depending on the current value of the PLC output MODE. If MODE is zero the station is in Automatic mode and if it is non-zero the station is in Manual mode.

The following messages can be used by Open Protocol to supervise the stations with respect to its Manual/Automatic state:

- MID 0400, “Automatic/Manual mode subscribe”

- MID 0401, “Automatic/Manual mode upload”
- MID 0402, “Automatic/Manual mode upload acknowledge”
- MID 0403, “Automatic/Manual mode unsubscribe”
- MID 0410, “AutoDisable settings request”
- MID 0411, “AutoDisable settings reply”

In addition to the “Automatic/Manual mode" messages the "Parameter set selected" (MID 0015) message can be used to keep track of the stations Automatic/Manual mode state. PowerMACS sends this message whenthe stations mode slection is changed, also when it changes to/from zero.

When entering Manual mode the following the following actions are executed internally by the PowerMACS station:

- The Open Protocol mode selection is zeroed.
- The VIN number is cleared.
- The station is enabled.
- The OK, or batch, counter is zeroed and turned off (not used in Manual mode).

If the connection to Open Protocol is lost while in Automatic mode then the following actions are executed:

- The Open Protocol mode selection is zeroed.
- The station is disabled.

8.11.1.2 Supported message types

MID	Name	Implementation description
0001	Communication start	Enables communication
0002	Communication start acknowledge	
0003	Communication stop	Disables communication
0004	Command error	
0005	Command accepted	
0010	Parameter set numbers upload request	Modes are used for parameter sets in PowerMACS.
0011	Parameter set numbers upload reply	
0014	Parameter set “selected” subscribe	
0015	Parameter set “selected”	Creation/download time for program run by first bolt in selected mode. Mode number not 0 in start signal gives that mode no according to start signal is used. Mode number 0 in start signal makes the station use the mode number it has stored. When testbolt is run mode number is reset to the mode number used before testbolt, when the testbolt cycle is ready.
0016	Parameter set “selected” acknowledge	
0017	Parameter set “selected” unsubscribe	
0018	Select Parameter set	Selection of Mode. Mode number is sent to station. ModeNo NOT_DEFINED (-32768) (test bolt not mode view) is not reported as parameter set change and mode is reset to original mode in station when the test bolt cycle has run.
0040	Tool data upload request	
0041	Tool data upload reply	Tool serial number (Id 01) is reported as spaces ' '. As Tool number of tightening (Id 02) is reported the number of cycles run on the spindle controller of the TC holding the epHan device (max value is 2147483647). As Last calibration date (Id 03) is reported the time for when the service flag was last reset (cycle count changes). Controller serial number (Id 04) is reported as spaces ' '.
0042	Disable tool	Disables START signal in PLC. When the station controller receives a START signal (from PLC or testbolt) a info event is issued to report that the station is disabled if a Disable tool signal has been received from the epHan device. When a Disable tool signal is received during a cycle an emergency stop is issued and the cycle stops. Emergency stop event is reported.
0043	Enable tool	Enables START signal in PLC. Station controller is enabled at startup. To enable the station controller when disabled an Enable tool signal must be sent.

MID	Name	Implementation description
0050	Vehicle Id Number download request	Selection of Wp Id. Wp Id is sent to PLC and CurrentIdString is updated the CurrentIdString is also reported to the station, which reports the Id to ephan device. When TestBolt is run Id is reset to the prior value after the cycle.
0051	Vehicle Id Number upload subscribe	
0052	Vehicle Id Number upload	
0053	Vehicle Id Number upload acknowledge	
0054	Vehicle Id Number upload unsubscribe	
0070	Alarm subscribe	Events mapped on 3-digit numbers. All events for the station and all general events are reported.
0071	Alarm Upload	There is no Controller ready status and Tool ready status available in the PowerMACS events. Status is always reported as OK.
0072	Alarm Upload acknowledge	
0073	Alarm Unsubscribe	
0074	Alarm Acknowledged on torque controller	Acknowledges all events in PM regardless of specified error number. TC sends this message when an event (subscribed on) has been marked observed in PM. If command "mark all as observed" is used event code 0 is reported.
0075	Alarm Acknowledged Ack	
0076	Alarm Status	Alarm status reports the oldest not already reported event. After alarm status Alarm Upload is done automatically until there are no not reported events in the queue. Alarm status 1 if any not observed events exist.
0077	Alarm Status acknowledge	
0080	Read time upload request	
0081	Time upload reply	
0082	Set Time in the Torque Controller	Only possible to perform on devices on the System TC.
0090	Multi spindle status subscribe	
0091	Multi spindle status upload	See TC Status Record Formats for message structure.
0092	Multi spindle status upload acknowledge	
0093	Multi spindle status unsubscribe	
0100	Multi spindle result subscribe	
0101	Multi spindle result upload	See TC Result Record Formats for message structure.
0102	Multi spindle result upload acknowledge	
0103	Multi spindle result unsubscribe	

MID	Name	Implementation description
0105	Last PowerMACS tightening result data subscribe	See document Open Protocol MID for PowerMACS data for more information
0106	Last PowerMACS tightening result Station data upload reply	See document Open Protocol MID for PowerMACS data.for more information
0107	Last PowerMACS tightening result Bolt data upload reply	See document Open Protocol MID for PowerMACS data for more information
0108	Last PowerMACS tightening result data acknowledge	See document Open Protocol MID for PowerMACS data for more information
0109	Last PowerMACS tightening result data unsubscribe	See document Open Protocol MID for PowerMACS data for more information
0240	User data download	See document Open Protocol MID for PowerMACS data for more information
0241	User data subscribe	See document Open Protocol MID for PowerMACS data for more information
0242	User data upload	See document Open Protocol MID for PowerMACS data for more information
0243	User data upload acknowledge	See document Open Protocol MID for PowerMACS data for more information
0244	User data upload unsubscribe	See document Open Protocol MID for PowerMACS data for more information
0400	Automatic/Manual mode subscribe	
0401	Automatic/Manual mode upload	
0402	Automatic/Manual mode upload acknowledge	
0403	Automatic/Manual mode unsubscribe	
0410	AutoDisable settings request	
0411	AutoDisable settings reply	
9999	Keep alive message	

8.11.1.3 TC Status Record Formats

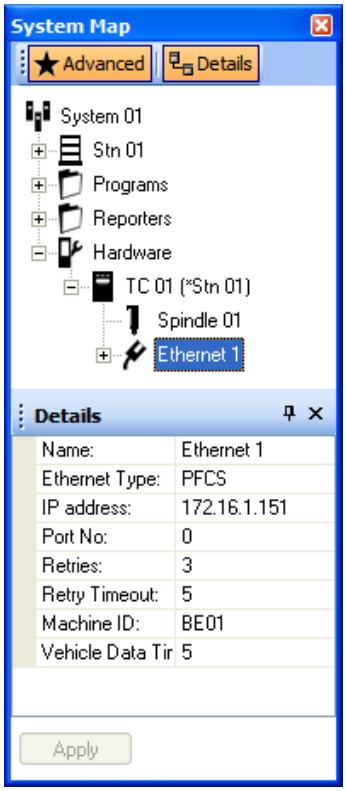
Parameter	Id	Bytes	Implementation
Number of spindles	01	2	The number of spindles is two bytes long and specified by 2 ASCII digits. Reports number of bolts in the cycledata. For PowerMACS 0 and 1 are possible numbers. In the case of 0 bolts in the cycledata the spindle specific part is omited (18 x 0 = 0).
Sync tightening Id	02	5	The sync tightening Id is a unique Id for each sync tightening result. Each individual result of each spindle is stamped with this Id. The tightening Id is incremented after each sync tightening. 5 ASCII digits. Max 65 535. Reports Station DataNo, created when cycle is ready.
Time	03	19	YYYY-MM-DD:HH:MM:SS
Sync overall status	04	1	Reports station status. The sync overall status is specified by one ASCII digit 1 = OK or OKR, 0 = NOK, TERMNOK or NOKRM.
Spindle status	05	5 × Number of spindles	Bytes 1-2: Ordinal bolt number (01 –99) Bytes 3-4: Spindle number (01-99) Byte 5: Reports 0 when bolt status is NOK, NOKRM or TERMNOK . Reports 1 when bolt status is OK or OKR.

8.11.1.4 TC Result Record Formats

Parameter	Id	Bytes	Implementation
Number of spindles	01	2	The number of spindles is two bytes long and specified by 2 ASCII digits. Reports number of bolts in the cycledata. For PowerMACS 0 and 1 are possible numbers. In the case of 0 bolts in the cycledata the spindle specific part is omited (18 x 0 = 0).
VIN Number	02	25	The VIN number is 25 byte long and is specified by 25 ASCII characters taken between 0x20 and 0x7F Hex. Reports Id string (CD_ST_IDDEV).
Job Number	03	2	The job number is two bytes long and always 00 as no such number is available in PowerMACS.
Pset number	04	3	This is the pset number that is run (psetId). The pset number is three byte long specifying a range of 000 to 999 and is specified by three ASCII digits (‘0’….’9’). Reports mode number.
Batch Size	05	4	The batch size is four byte long and always 0000 as no such number is available in PowerMACS.
Batch counter	06	4	The batch counter number is four byte long and always 0000 as no such number is available in PowerMACS.
Batch status	07	1	The batch status is specified by one ASCII character. Always 0 as no such number is available in PowerMACS.
Torque Min limit	08	6	The torque min limit is multiplied by 100 and sent as an integer (2 decimals truncated). The torque min limit is six byte long and is specified by six ASCII digits (‘0’….’9’). Bolt T, Min limit, from the Monitor function. Value retrieved from the first bolt reported in the cycle data. Reports measured value in global torque unit times 100.
Torque Max limit	09	6	Reports Bolt T, Max limit, from the Monitor function. Same format as Torque Min limit is reported in.
Torque final target	10	6	Bolt T from the Monitor function, which is the measured final torque in the monitor function. Same format as Torque Min limit is reported in.
Angle Min	11	5	The angle min value has a specified range between 0 and 99999. The angle min value is five byte long and is specified by five ASCII digits (‘0’….’9’). Bolt A, Min limit, from the Monitor function. The value is retrieved from the first bolt reported in the cycle data
Angle Max	12	5	Bolt A, Max limit, from the Monitor function. Same format as Angle Min limit is reported in.
Final Angle Target	13	5	Bolt A from the Monitor function, which is the measured angle in the monitoring function. Same format as Angle Min limit is reported in.
Date/time of last change in Pset settings	14	19	Time stamp for the last change in the current program or time for latest complete download. The time stamp is 19 byte long and is specified by 19 ASCII characters (YYYY-MM-DD:HH:MM:SS).

Parameter	Id	Bytes	Implementation
Time	15	19	YYYY-MM-DD:HH:MM:SS
Sync tightening Id	16	5	The sync tightening Id is a unique Id for each sync tightening result. Each individual result of each spindle is stamped with this Id. The tightening Id is incremented after each sync tightening. 5 ASCII digits. Max 65 535. Reports Station DataNo, created when cycle is ready.
Sync overall status	17	1	Reports 1 when station status is OK or OKR. Reports 0 when station status is NOK, TERMNOK or NOKRM.
Spindle status	18	18 × Number of spindles	Bytes 1-2: Ordinal bolt number (01 –99) Bytes 3-4: Spindle number (01-99) Byte 5: Reports 0 when bolt status is NOK, NOKRM or TERMNOK. Reports 1 when bolt status is OK or OKR. Byte 6: Reports 0 if bit CD_ERR_FTLM set for ErrorCode. Equally 2 is reported if FTHM set. If neither FTLM nor FTHM is set 1 is reported Bytes 7-12: Bolt T from the Monitor function for current bolt. Same format as Torque Min limit is reported in. Byte 13: Equal to Byte 6, but looks at bit ALM and AHM. Byte 14-18: Bolt A from the Monitor function for current bolt. Same format as Angle Min limit is reported in.

8.11.2 DaimlerChrysler PFCS



Specify in **Port No** which port to use in the server.

Retries specifies how many retries should be done in case a telegram has not been able to send. **Retry Timeout** specifies the timeout between retries.

The **Machine ID** is the identification that is sent first in all telegrams to the server and which identifies the source of all telegrams.

Vehicle Data Timeout is the timeout used when transmitting Vehicle Data from the server to PowerMACS.

In the text blow the server is described as PFCS (Plant Floor Communication System) and the PowerMACS as PFD (Plant Floor Device),

The protocol is implemented according to the Daimler Chrysler document "PFCS Vendor Specifications", revision no 4.

8.11.2.1 Supported message types

Following message types are supported:

- 0002: Test Results to PFCS from PFD
- 9999: Keep alive message from PFD

PFCS can also send a 0003 message, unsolicited vehicle data, to PFD. PFD will accept the message and return an ACK message but will currently not perform any actions.

Please see chapter Cycle Data Storage for more information about saving cycle data.

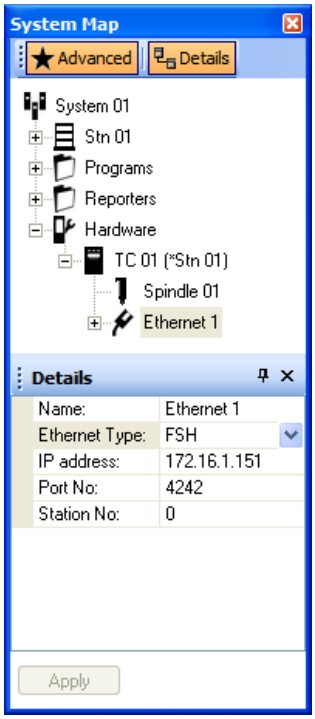
8.11.2.2 PFD Result Record Formats

The tightening results are reported to PFCS according to the specification, with following additional information.

Field	**Implementation**
VIN	Last 6 characters of the ID string
Track Seq No	Always 000000
Spindle Number	Bolt number
Bolt Count	Bolt number
Torque High Limit	Torque high limit from the Monitor function
Torque Low Limit	Torque low limit from the Monitor function
Torque Reading	Torque value the Monitor function
Angle High Limit	Angle high limit from the Monitor function
Angle Low Limit	Angle low limit from the Monitor function
Angle Reading	Angle value the Monitor function

8.11.3 FSH

FSH is a protocol specified by General Motors used to communicate between the PowerMACS and the Fastening System Host (FSH)



Specify in **IP address** the TCP/IP address of the Fastening system Host (FSH).

Specify in **Port No** which port to use in the FSH. (default is 4242)

The **Station No** is the identification of the TC in the FSH, this is sent in all telegrams and identifies the source of all telegrams.

In the following text the server is described as FSH (Fastening system Host) and the PowerMACS as TC (Torque controller),

The protocol is implemented according to the General Motors document "GAE Electric Nutrunner Controller Design Criteria", GAEc-03 6-9-03Revision.doc.

8.11.3.1 Supported message types

The following message types are supported:

Packet number	Messages
1	FSH Station ID No Request
2	TC Station ID No Acknowlwdge
3	FSH Rundown Packet Request
4	TC Rundown Packet
5	FSH Rundown Packet Acknowledge
6	TC No Rundown Packet
7	FSH Reset Buffer Request
8	TC Reset Buffer Acknowlwdge
15	FSH Communication Parameters Packet
16	TC Communication Parameters Packet Acknowledge
17	FSH Set Date and Time Packet
18	TC Set Date and Time Packet Acknowledge

Please see chapter Cycle Data Storage for more information about saving cycle data.

8.11.3.2 TC Result Record Formats

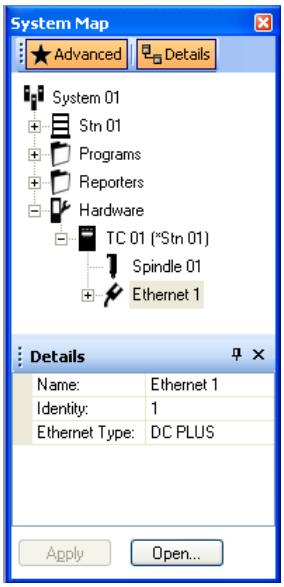
The tightening results are reported to FSH according to the specification, with following additional information.

Field	Implementation
Network Id	IP address of TC.
Station Number	Configurable via ToolsTalk PowerMACS.
Cycle Number	Data Number.
Number of Spindles	Reports number of bolts in the cycle data.
Parameter Set	Mode number
Supplier Code	"ACS".
Vehicle ID code. (PVI No)	ID device string converted to a number.
Spindle	Ordinal bolt number within the station.
Torque	Bolt Torque from the Monitor function.
Torque Low Limit	Bolt Torque, Min limit, from the Monitor function.
Torque High Limit	Bolt Torque, Max limit, from the Monitor function.
Angle	Bolt Angle from the Monitor function.
Angle Low Limit	Bolt Angle, Min limit, from the Monitor function.
Angle High Limit	Bolt Angle, Max limit, from the Monitor function.
Status Bits	Completion, Torque Status, Torque Spec, Angle Status, and Angle Spec are set in accordance to the corresponding bits in the Bolt level result variables "Compact Errors". The Time Status and Time Spec bits are set according to the status of the bits TIR, TIL, TIH in the Bolt level result variables "Errors". The following rules are used:: • If neither TIR, TIL nor TIH are set then Time Status and Time Spec are set to 0. • If at least one of TIR, TIL or TIH is set then Time Status is set to 1. If TIL is set then Time Spec is set to 0 (regardless if TIR and/or TIH are set as well). If only TIR and/or TIH are set then Time Spec is set to 1.
Tool Serial Number	The value of the Spindle Set Up/General parameter "Spindle Serial Number" for the spindle that was used to tighten the bolt in question.

8.11.4 DC PLUS

The PLC variables used to control this device are described in chapter DC PLUS variables.

PowerMACS supports two different versions of the DC PLUS, **PDT** and **PQD**



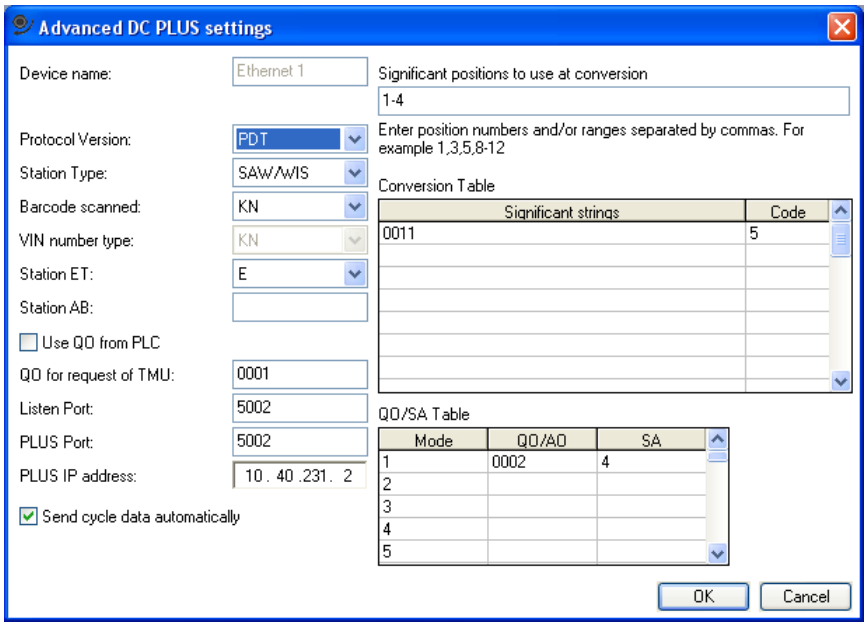
The PLC variables used to control this device are described in chapter DC PLUS variables.

Press the **Open...** button on the **Details** form to to open the Advanced DC PLUS settings form.

The different parameters and their function are described below.

Protocol Version – There are two different versions of the protocol, PDT and PQD. The parameters needed for the device vary a little depending on the selected version.

8.11.4.1 Parameters for PDT



Station Type - There are four different station types, SAW/WIS, SPS, VBA and SSE. Controls the behavior of the device. See the description of the different types below.

Barcode scanned - Possible choices are LI, PN and KN. Specifies the type of the scanned barcode.

VIN number type -Possible choices are PN or KN. Controls how "Wp ID" shall be treated when sending cycle data to PLUS. Only possible to select if “Barcode scanned” is LI, otherwise the value is the same as “Barcode scanned”.

Station ET -Possible choices are E and G. The value selected here is used as “Einzelergennis Typ Kz” when sending tightening results to PLUS.

Station AB -The value to use as AB “Anlagen betreiber” for all bolts when sending cycle data to PLUS. Will also be included in the cycle data as the result variable Station AB.

Use QO from PLC -Controls which value to use as QO when sending requests for TMU. If checked the value is taken from the PLC, otherwise the value “QO for request of TMU” will be used.

QO for request of TMU -The value to use as QO when sending a request for a TMU to the PLUS server. Only enabled if “Use QO from PLC” is unchecked and “Station Type” is SAW/WIS.

Listen Port -The port that PM listen to for PLUS connections. Valid port numbers are 20 000 – 32 767

PLUS IP Address -The IP address of the PLUS server to communicate with.

PLUS Port -The port in the PLUS server that PowerMACS connects to. Valid port numbers are 0 – 32 767.

Send cycle data automatically -Controls the sending of cycle data. If checked the result from a tightening will be sent to PLUS immediately after the cycle has finished.

If the connection to the PLUS server is broken when the cycle is finished the data will be sent when the connection is established again.

If unchecked the sending of data is controlled by the PLC variable PLUS_DOKUENABLE instead.

Conversion table -Converts the received TMUs to a mode number. Has the same functionality as the conversion tables used for the Stations Work piece identifier and Multiple identifiers functions.

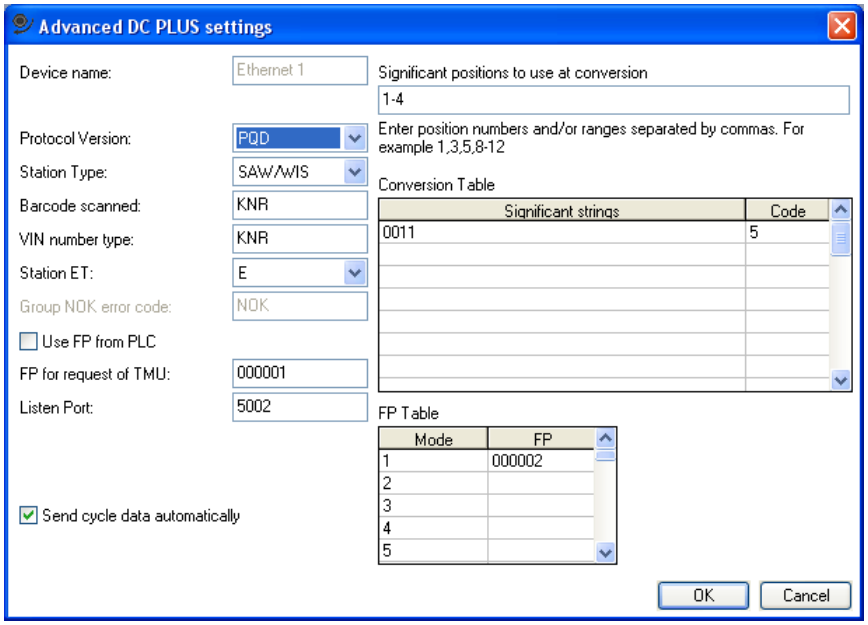
QO/SA table -The value to use as QO, “Qualitätsaussage Orts Nr”, and SA, “Soll anzahl Einzelergebnisse” when sending cycle data to PLUS is dependent on the mode used for the tightening. The QO and SA to use are specified in this table with one row for each mode. If no values are specified for a mode, cycle data produced when running this mode will not be sent to the PLUS server.

The values specified will also be included in the cycle data as the result variables Station QO and Station SA.

The value specified as QO will also be used as AO, “Anlagen Orts Nr”, for all bolts in the cycle data.

Test PLUS -When this button is pressed the connection to the PLUS server is tested. The result of the test will be indicated as OK or NOK. The test button is only available if the advanced form is opened connected to the target system.

8.11.4.2 Parameters for PQD



Station Type - There are four different station types, SAW/WIS, SPS, VBA and SSE. Controls the behavior of the device. See the description of the different types below.

Barcode scanned - Specifies the type of the scanned barcode.

VIN number type - Controls how "Wp ID" shall be treated when sending cycle data to PLUS. Only possible to select if “Station Type” is SAW/WIS, otherwise the value is the same as “Barcode scanned”.

Station ET -Possible choices are E and G. Controls how OK bolts in cycles that are NOK shall be reported to PLUS. With the selection E all bolts keep there original status, the NOK bolts are reported as NOK and the OK bolts as OK. With the selection G ALL bolts are reported as NOK, even the bolts that was really OK. For the originally OK bolts the string specified in **Group NOK error code** will be used in the field FE, “Fehlercode Anlage”, to indicate that the bolt status was changed.

Use FP from PLC -Controls which value to use as FP when sending requests for TMU. If checked the value is taken from the PLC, otherwise the value “FP for request of TMU” will be used.

FP for request of TMU -The value to use as FP when sending a request for a TMU to the PLUS server. Only enabled if “Use FP from PLC” is unchecked and “Station Type” is SAW/WIS.

Listen Port -The port that PM listen to for PLUS connections. Valid port numbers are 4950 – 5050 and 20 000 – 32 767

Send cycle data automatically -Controls the sending of cycle data. If checked the result from a tightening will be sent to PLUS immediately after the cycle has finished.

If the connection to the PLUS server is broken when the cycle is finished the data will be sent when the connection is established again.

If unchecked the sending of data is controlled by the PLC variable PLUS_DOKUENABLE instead.

Conversion table -Converts the received TMUs to a mode number. Has the same functionality as the conversion tables used for the Stations Work piece identifier and Multiple identifiers functions.

FP table -The value to use as FP, “Fertigungspunkt” when sending cycle data to PLUS is dependent on the mode used for the tightening. The FP to use are specified in this table with one row for each mode. If no values are specified for a mode, cycle data produced when running this mode will not be sent to the PLUS server.

The values specified will also be included in the cycle data as the result variables Station QO

Test PLUS - When this button is pressed the connection to the PLUS server is tested. The result of the test will be indicated as OK or NOK. The test button is only available if the advanced form is opened connected to the target system.

8.11.4.3 Handling of PN, KN and LI, in protocol version PDT

The handling of PN, KN and LI are controlled by the two parameters **Barcode scanned** and **VIN number type**.

Request for TMU

When PowerMACS requests a TMU from the PLUS server the value to use for PN, KN or LI is sampled from the PLC variable PLUS_BARCODE. If the value shall be treated as a PN, KN or LI is controlled by **Barcode scanned**, see table below.

Barcode scanned	VIN number type	PLUS_BARCODE is treated as	Telegram sent to PLUS
PN	PN	PN	NSR ADAA
KN	KN	KN	NSR KDAA
LI	PN	LI	NSR ALAA
LI	KN	LI	NSR KLAA

Sending results

When PowerMACS sends result data to the PLUS server the value to use for PN or KN is sampled from the Wp ID field in the result. **VIN number type** controls if Wp ID shall be treated as PN or KN, see also the table below.

Barcode scanned	VIN number type	Wp ID is treated as:	Telegram sent to PLUS
PN	PN	PN	ISR ADEA
KN	KN	KN	ISR KDEA
LI	PN	PN	ISR ALEA
LI	KN	KN	ISR KLEA

The Wp ID shall always begin with the letters “PN” or “KN” if everything is correct. These letters are removed before the result is sent to PLUS.

Before the cycle data is sent Wp ID is checked for consistency. If it begins with “PN” and “VIN number type” is KN or vice versa an event is generated. The same will also happen if the letters “PN” or “KN” is completely missing or if the length of Wp ID is not correct. If the layout of Wp ID is wrong no data will be sent to PLUS.

8.11.4.4 Handling of QO and FP

QO and FP is the same thing, it is only the name that differ, and they are handled in the same way in both protocol versions. In protocol version PDT the name is QO and in PQD the name is FP.

The value that PowerMACS uses as QO “Qualitätsaussage Orts Nr” is controlled by the parameters **QO for request of TMU**, **Use QO from PLC** and the **QO/SA table**. The PLC system global **PLUS_QO** is also used.

Request for TMU

When PowerMACS asks for a TMU the value written in **QO for request of TMU** is used as QO if **Use QO from PLC** is unchecked.

If **Use QO from PLC** is checked the value of the PLC system global PLUS_QO is used instead. If the value of PLUS_QO is not a valid QO an event is generated to inform the user and no request for TMU is sent.

Use QO from PLC	Value used as QO
Checked	PLUS_QO in the PLC
Unchecked	QO for request of TMU

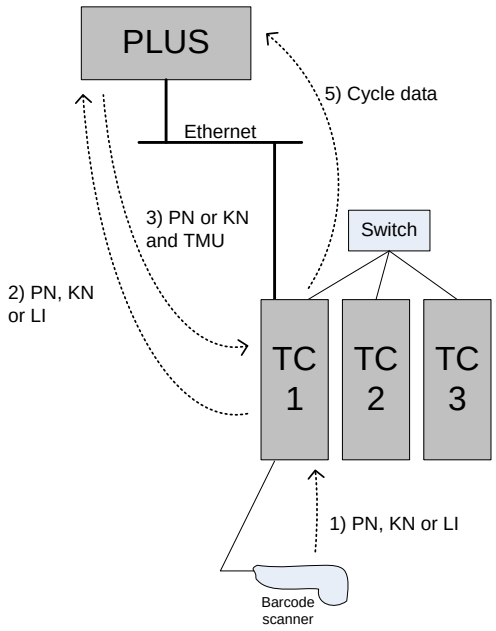
Sending results

When PowerMACS send tightening results to PLUS the value to use as QO is taken from result variable Station QO stored in the cycle data.

The value to store in Station QO is determined by the mode number, according to the **QO/SA table**. The first mode that was started will decide the QO. In the case of stitching different modes can be used for one tightening, but the value will always be from the first mode.

8.11.4.5 SAW / WIS

In this station type, a barcode scanner is connected directly to the Station TC.



Workflow

1. The PN, KN or LI number is read by the barcode scanner, or automatic scanner, and sent to the TC.
2. The TC sends a request for TMU to PLUS with the PN, KN or LI that was scanned.
3. PLUS sends the TMU and PN or KN to use for the tightening to the TC.
4. The TC converts the TMU to a mode number and makes the tightening.
5. When the tightening is finished the cycle data is sent to PLUS.

PLC program

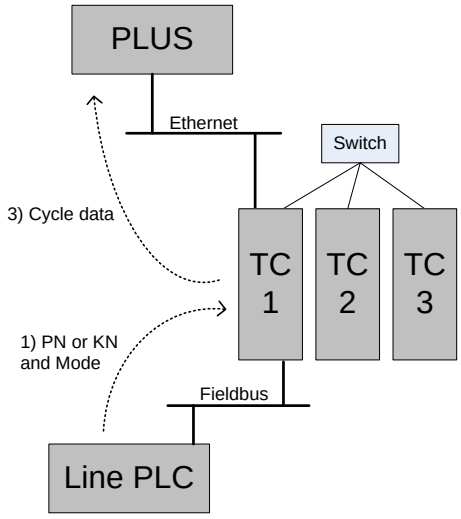
The PowerMACS PLC must handle the string read from the barcode and request a TMU from PLUS.

1. The barcode is scanned and the value is put in CURRIDSTRING in the PowerMACS PLC.
2. The PowerMACS PLC copies the value in CURRIDSTRING to PLUS_BARCODE and sends a positive edge to PLUS_GETTMU.
3. When the PLUS server has sent an answer PLUS_GETTMU_OK will be set to TRUE.

   PLUS_MODE contains the mode number to use for the tightening. The received TMU is converted to the mode number using the conversion table for the device. CURRIDSTRING contain the PN or KN sent by PLUS with the letters “PN” or “KN” added as prefix.
4. PLUS_MODE is copied to MODE by the PowerMACS PLC. This mode is started and CURRIDSTRING will be included as Wp. Id in the cycle data.

8.11.4.6 SPS

In this station type, an external PLC controls the station. The TC gets the VIN number and mode number from the external PLC over a field bus connection.



Workflow

1. The external PLC sends a PN or KN and a mode number over the field bus.
2. The TC makes the tightening with the selected mode.
3. When the tightening is finished the cycle data is sent to PLUS.

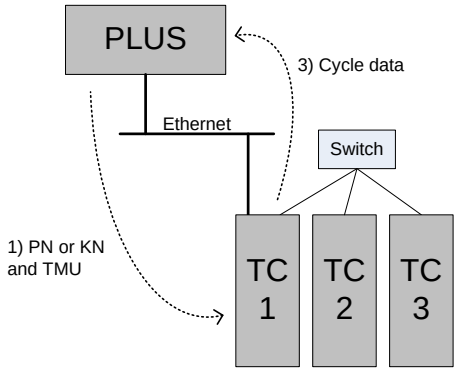
PLC program

The PowerMACS PLC program will be very similar to the program used in the SAW station.

1. The line PLC sends mode number and PN or KN to PowerMACS PLC through a field bus connection.
2. The PowerMACS PLC copies the value of PN or KN to PLUS_BARCODE and sends a positive edge to PLUS_GETTMU.
3. Since **Station type** is SPS nothing is sent to PLUS. Instead the value written in PLUS_BARCODE is directly copied to CURRIDSTRING with a “PN” or “KN” added as prefix.
4. The mode number received from the line PLC is copied to MODE, by the PowerMACS PLC. This mode is started and CURRIDSTRING will be included as Wp. Id in the cycle data.

8.11.4.7 VBA

In this station type the VIN number and TMU are sent directly from PLUS, without a request from the TC.



Workflow

1. PLUS sends PN or KN and a TMU to the TC.
2. The TC converts the TMU to a mode number and makes the tightening.
3. When the tightening is finished the cycle data is sent to PLUS.

PLC program

Normal workflow:

1. PLUS sends a PN or KN and a TMU to the TC. The PN or KN is automatically copied to CURRIDSTRING with a “PN” or “KN” added as prefix. The TMU is automatically converted to a mode number and the result is placed in PLUS_MODE.
2. PLUS_MODE is copied to MODE by the PowerMACS PLC. This mode is started and CURRIDSTRING will be included as Wp. Id in the cycle data.

NSCR:

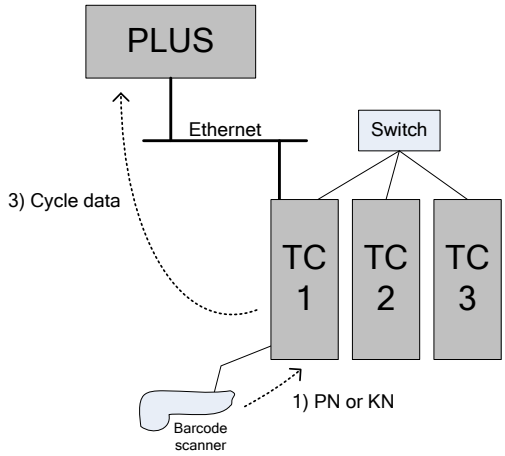
If PLUS sends the telegram NSCR it means that no tightening shall be made for the PN or KN and TMU that was just received. This is indicated to the PLC by clearing the CURRIDSTRING and PLUS_MODE. If this happens the PLC program must recognize this and not start a tightening.

Automatic and Manual mode:

The VBA station can operate in Automatic or Manual mode. Automatic mode is the normal case and is the one that is describe in the picture above. In manual mode no connection to PLUS is needed, the PLC receives the mode number from some other device and no cycle data shall be sent to PLUS. The PowerMACS PLC must inform the TC that the station is running in manual mode with the PLC variable PLUS_MANUALMODE to make sure no data is sent.

8.11.4.8 SSE

In this station type there are no TMU, instead the PN or KN is decoded to get the mode number.



Workflow

1. The PN or KN number is read by the barcode scanner and sent to the TC.
2. The TC converts the PN or KN to a mode number and makes the tightening.
3. When the tightening is finished the cycle data is sent to PLUS.

PLC program

The PowerMACS PLC program will be very similar to the program used in the SAW station.

1. The barcode is scanned and the value is put in CURRIDSTRING in the PowerMACS PLC.
2. The PowerMACS PLC copies the value in CURRIDSTRING to PLUS_BARCODE and sends a positive edge to PLUS_GETTMU.
3. Since **Station type** is SSE nothing is sent to PLUS. Instead the value written in PLUS_BARCODE is directly copied to CURRIDSTRING with a “PN” or “KN” added as prefix.

   The string in PLUS_BARCODE is also sent to the conversion table for the device and the result is placed in PLUS_MODE.
4. PLUS_MODE is copied to MODE by the PowerMACS PLC. This mode is started and CURRIDSTRING will be included as Wp. Id in the cycle data.

8.11.4.9 TC Result Formats for PDT

The tightening results are reported to PLUS according to the specification.

Station data:

Field		Implementation
QUALITÄTSAUSSAGE-ORTS-NR	QO	Station QO in the cycle data.
JOB-NR	JN	Mode number for the tightening.
GESAMT-Q-AUSSAGE	GQ	Station status of a tightening. 1 = OK is sent for status OK and OKR (OK after repair) 0 = NOK is sent for status NOK and TERMNOK (failed during termination).
GESAMT-STATUS	GS	Always 0
SOLL-ANZAHL-EINZELERGEBNISSE	SA	Station SA in the cycle data.
EINZELERGEBNIS-TYP-KZ	ET	The value in **Station ET.**

Bolt data:

Field		Implementation
EINZELERGEBNIS-NR	EN	Bolt number from ToolsTalk PowerMACS setup.
EINZELERGEBNIS-IDENT	EI	Bolt name from ToolsTalk PowerMACS setup. The bolt number will be added at the end of the bolt name to reach 24 characters.
EINZELERGEBNIS-Q-AUSSAGE	EQ	Bolt status for this tightening 1 = OK is sent for status OK and OKR (OK after repair) 0 = NOK is sent for status NOK, TERMNOK (failed during termination) and NOKRM (not successful due to reject management)
EINZELERGEBNIS-STATUS	ES	Always 0
EINZELERGEBNIS-QA-DETAIL	ED	Customer Error Code in the cycle data.
ANLAGEN-ORTS-NR	AO	Will always be the same as QO for all bolts.
ANLAGEN-BETREIBER	AB	Station AB in the cycle data, always the same for all bolts.
USER-ID	US	Id Res 4 in the cycle data, always the same for all bolts.

Result data:

Bolt level data

This table describes the data sent to PLUS if Bolt level data is reported

PI	Description	Value sent, PW	Comment
001	Torque	Bolt T	The value reported from “Final Torque at cycle end” monitoring. The torque is multiplied by 100 and is sent as an integer (2 decimals rounded).
002	Angle of rotation	Bolt A	The value reported from “Angle from threshold to Cycle End” monitoring. Are always sent as an integer with unit degrees. The value is rounded to nearest integer.
003	Step number	0	The step number will always be 0 to indicate that it is bolt data

Step level data

This table describes the data sent to PLUS if step level data is reported. All the step data will come from the same step.

PI	Description	Value sent, PW	Comment
001	Torque	SO T, “Shut off torque”	The value reported from a “Check shut off torque” check. The torque is multiplied by 100 and is sent as an integer (2 decimals rounded).
002	Angle of rotation	A	The value reported by a “Check angle” check. Are always sent as an integer with unit degrees. The value is rounded to nearest integer.
003	Step number	Step No	The number of the step that produced the other data.
004	Current	Peak C	The highest current reached during the step. Reported by a “Check peak torque” check running on channel T3. The current is converted to torque, which is then multiplied by 100 and sent as an integer (2 decimals rounded).

8.11.4.10 TC Result Format for PQD

The tightening results are reported to PLUS according to the specification.

Station data:

Field		Implementation
Fertigungspunkt	FP	Station QO in the cycle data
Id-Typ	IT	The string specified as VIN number type
Id-Wert	IW	Wp. Id. In the cycle data
Q-AUSSAGE-ART	AA	Always ‘SC’

Bolt data:

Field		Implementation
Q-AUSSAGE-IDENT	QI	Bolt Id from WinTC setup
Q-AUSSAGE	QA	Bolt status for this tightening IO = OK is sent for status OK and OKR (OK after repair) NIO = NOK is sent for status NOK, TERMNOK (failed during termination) and NOKRM (not successful due to reject management)
FEHLERCODE-ANLAGE	FE	Customer Error Code in the cycle data
Q-AUSSAGE-STATUS	QS	Always 00
USER-ID	US	Id Res 4 in the cycle data, always the same for all bolts.

Result data:

The following data is possible to report to DC PLUS. By default only the Torque, Angle, Redundancy Torque and the Step number is included. The other data can be selected in the reporter if wanted.

Bolt level data

This table describes the data sent to PLUS if the Bolt level data is reported

PI	Description	Value sent, PW	Comment
DM	Torque	Bolt T	
MO	Torque high limit	Bolt T, high limit	Only sent if limits for Bolt T is selected in the reporter
MU	Torque low limit	Bolt T, low limit	Only sent if limits for Bolt T is selected in the reporter
RM	Redundancy torque	-	No value is sent
DW	Angle	Bolt A	Are always sent as an integer with unit degrees. The value is rounded to nearest integer.
WO	Angle high limit	Bolt A, high limit	Only sent if limits for Bolt A is selected in the reporter
WU	Angle low limit	Bolt A, low limit	Only sent if limits for Bolt T is selected in the reporter
RW	Redundancy angle	-	This value is not possible to send.
ZW	Time	-	No value is sent
PH	Step number	0	The step number will always be 0 to indicate that it is bolt data
ST	Current	-	No value is sent
GD	Gradient	-	No value is sent

Step level data

This table describes the data sent to PLUS if step level data is reported. All the step data will come from the same step.

PI	Description	Value sent, PW	Comment
DM	Torque	SO T, “Shut off torque”	The value reported from a “Check shut off torque” check. The torque is multiplied by 100 and is sent as an integer (2 decimals rounded).
MO	Torque high limit	SO T, high limit	Only sent if limits for SO T is selected in the reporter
MU	Torque low limit	SO T, low limit	Only sent if limits for SO T is selected in the reporter
RM	Redundancy torque	Peak C	The current is converted to torque.
DW	Angle	A	Are always sent as an integer with unit degrees. The value is rounded to nearest integer.
WO	Angle high limit	A, high limit	Only sent if limits for A is selected in the reporter

WU	Angle low limit	A, low limit	Only sent if limits for SO T is selected in the reporter
RW	Redundancy angle	-	This value is not possible to send.
ZW	Time	Time	The time that the step was run, measured by the Check time check. Only sent if Time is selected in the reporter
PH	Step number	Step No	The step number for the step that produced the other data
ST	Current	-	Not sent, the current is used as redundancy torque, RM, instead.
GD	Gradient	Grad	The value produced by the Gradient Restriction. Only sent if Grad is selected in the reporter

8.11.4.11 Selecting the data to report

Only one angle value and one torque value can be sent for each tightening. Bolt T and Bolt A are reported in OK cycles and SO T and A (from the failing step) in NOK cycles.

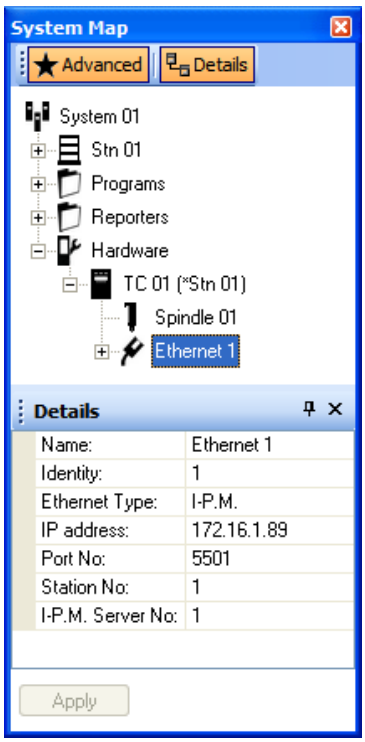
If Bolt level data is reported but both Bolt T and Bolt A are empty the step data for the last step with both SO T and A is reported instead.

8.11.4.12 Selecting the data to report

Only one angle value and one torque value can be sent for each tightening. Bolt T and Bolt A are reported in OK cycles and SO T and A (from the failing step) in NOK cycles.

If Bolt level data is reported but both Bolt T and Bolt A are empty the step data for the last step with both SO T and A is reported instead.

8.11.5 I-P.M.



Specify in **IP address** the TCP/IP address of the I-P.M. server.

Specify in **Port No** which port to use in the I-P.M. (default is 5501)

Station No specifies the station number this station use to identify itself in telegrams sent to I-P.M.

Set **I-P.M. Server No** to the server number of the I-P.M. server (shall normally be 1).

8.11.5.1 Mapping of data

The I-P.M. implementation is done following version 3.1.11 of the I-P.M. specification with the following additional information.

Telegram Header

Field	Bytes	Value sent
Source	8	"AC.0001 ", the number is the station, number selected in the device in WinTC, numbers between 1 and 9999 are possible to use
Destination	8	"IPM.0001", the number is the IPM number selected in the device in WinTC, numbers between 1 and 9999 are possible to use
Identification	8	"IPM-DATA"

Field	Bytes	Value sent
Version	2	"3 "
Character Code	20	"ISO-8859-1 "
Sequence Number	4	
Length of User Data	6	
Byte Order	1	Always 0 = Intel byte order

Maintenance Sequence User Data

Field	Bytes	Value sent
Manufacturer Identification	5	"AC "
Facility Type	2	1 if Facility type = "Fastening" and 4 if = "Adjustment, for the moment always 1
Date	10	Cycle start time
Time	8	Cycle start time
Maintenance Sequence, AFU-number	30	See description below.
Event Number	15	Empty
Maintenance Sequence Text	50	"Bolt name" + "-" + "Program name" e.g. "Bolt 01-Test run", "Left bolt-Tighten left bolt"
Tool-ID	10	Spindle serial number from the spindle setup, if it is specified.
Unit Carrier ID	10	Result variable Id res 2, only the 10 first characters are used.
Type	10	Result variable Id res 3, only the 10 first characters are used.
Identification	30	Result variable Id res 1 or Wp ID, depending on the selection in the I-P.M. reporter, only the 30 first characters are used.
Additional ID	10	Result variable Id res 4, only the 10 first characters are used.
Curve	1	0 = without trace 1 = with trace
Status	2	0 = OK or OKR 1 = NOK, TERMNOK or NOKRM This is always the bolt status; a bolt can be OK even if the station is NOK.
Mode	1	Always 0 = automatic mode
Loosen	1	" " = not available.
Number of additional information	2	Always "0 ", no additional information is available
Number of Error Codes	2	
Error Code (for every error)	5	See table below for possible error codes

Field	Bytes	Value sent
Number of Characteristics	2	

Characteristic Data

Field		Bytes	Value sent
Identification		5	See table below for possible values
Unit		10	
Length		1	
Factor		2	
Algebraic Sign		1	
Increment		8	
Start Value		8	
Actual Value		8	The result variable
Step		2	Always 1
Number of Parameters		2	
For every parameter	Id	6	See table below for possible values
	Value	8	The value

8.11.5.2 Value sent as Maintenance Sequence (AFO-number)

The value sent in the field Maintenance Sequence is built up using the following result variables:

Station name
Spindle number
Bolt number
Program number

The layout of the maintenance sequence is controlled by the settings in the I-P.M. reporter.

The station name is put first in the maintenance sequence; the value of the field width in the reporter sets the number of characters to use. If width is empty the whole station name is used as it is.

After the station name the spindle number and bolt number are sent if they are selected in the reporter. The one that has the lowest order value selected in the reporter are sent first. They are added directly after the station name without any additional characters in-between

Last in the maintenance sequence the program number is sent, if it is selected in the reporter. Between the spindle number/bolt number and the program number the character '-' is added. This character is only added if a program number is sent. If the program that was used didn't have a program number specified, the number 0 is used.

For the values spindle number, bolt number and program number the setting in the field width in the reporter sets the number of digits to use, it is padded with 0's to the left to reach this number of digits. If width is empty the actual number of digits in the result value is used.

The string sent as Maintenance Sequence is left aligned and padded with spaces to reach 30 characters. If the total numbers of characters selected are more than 30 only the first 30 (counted from left) will be sent. In this case an event will be issued to inform the user of the occurrence.

Example: With the following settings in the reporter:

Result variable	Order:	Width:
Station name	1	6
Spindle number	2	3
Bolt number		
Program number	4	2

Station name = `Test1234`
Spindle Number = `12`
Bolt Number = `17`
Program Number = `21`

The maintenance sequence will be "`Test12012-21`"

8.11.5.3 Possible error codes

Value sent	Meaning	Corresponding PowerMACS error
1	MD-max	THM, torque high monitoring
2	MD-min	TLM, torque low monitoring
3	WI-max	AHM, angle high monitoring
4	WI-min	ALM, angle low monitoring
5	Angle different	DEV1M, DEV2M or DEV, Deviation in interval 1 or 2 to high during monitoring or Deviation to high from check. Set if any of these errors are present.
6	Gradient max	TRH, Torque rate to high from check.
7	Gradient min	TRL, Torque rate to low from check.
8	Draft1 max	TR1HM, torque rate 1 high monitoring
9	Draft1 min	TR1LM, torque rate 1 low monitoring
10	Draft2 max	TR2HM, torque rate 2 high monitoring
11	Draft2 min	TR2LM, torque rate 2 low monitoring
13	Not OK-Time	TIH or TIL, time high or low from check Also TIR, time restriction
18	Not enough WI	AL, angle low from check
19	Self tap max	PVTH, post view torque too high from check

Value sent	Meaning	Corresponding PowerMACS error
20	Current max	CH, current high from check
21	Current min	CL, current low from check
22	Security, MD	TR, torque restriction
23	Security, WI	AR, angle restriction
499	Other error	If any other error code is set.

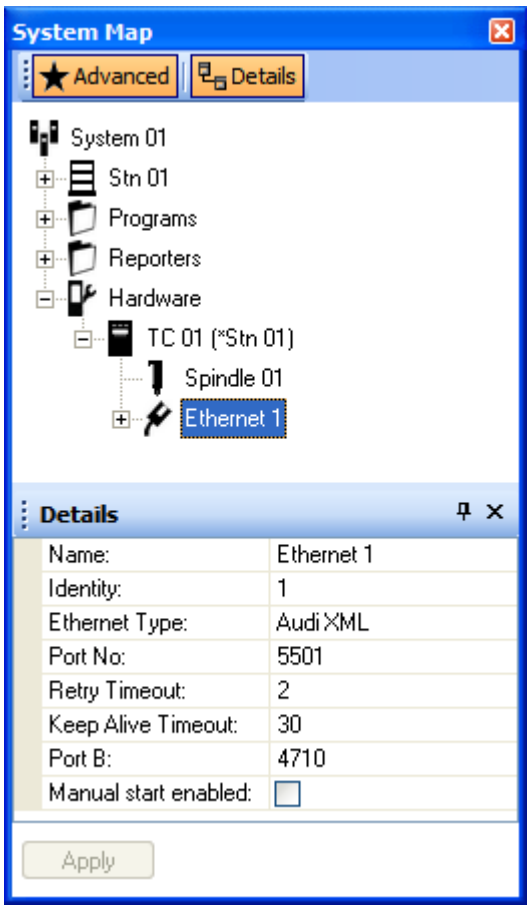
8.11.5.4 Characteristic Identifications

Identification	Meaning	Value sent	Unit
WI	Actual value angle	Bolt A	Grad
MI	Actual value torque	Bolt T	Nm
GI	Actual values gradient	Bolt TR1	Nm/Gr
TI	Actual value time	Only sent as trace data	S
MR	Redundancy Torque	Bolt A 2nd	Grad
WR	Redundancy angle	Bolt T 2nd	Nm

8.11.5.5 Parameter Identifications

Identification	Meaning	Value sent
W-	Lower tolerance angle	Bolt A low limit
W+	Upper tolerance angle	Bolt A high limit
M-	Lower tolerance torque	Bolt T low limit
M+	Upper tolerance torque	Bolt T high limit
G-	Lower tolerance gradient	Bolt TR1 low limit
G+	Upper tolerance gradient	Bolt TR1 high limit
MS	Threshold	Bolt A Thresh T

8.11.6 Audi XML



8.12 Fieldbus Interface

On a tightening controller it is possible to have a fieldbus slave interface module mounted. This makes it possible to interact with your PowerMACS system from a fieldbus master. A fieldbus device handles access via fieldbus.

Currently the following fieldbuses are supported:

- Profibus-DP and DP-V1
- DeviceNet
- Modbus TCP
- EtherNet/IP
- ProfiNet IO

Using a fieldbus device you can access the following PowerMACS functions:

- Read and write PLC signals
- Read cycle data
- Read events
- Read traces
- Read and write complete setups

From PowerMACS point of view most fieldbus types looks the same.

To access the PowerMACS system via a fieldbus you first have to add a fieldbus device, as described in chapter: Add a device, to TC that has a PLC, i.e. the first TC in a station.

A PowerMACS PLC can only access a fieldbus interface that is mounted on the same TC as the PLC is running on.

Note! The DeviceNet GM is a special case in the sense that its input and output maps are predefined and fixed. The inputs and outputs of the interface is not accessible from the PowerMACS PLC but are handled by the PowerMACS station controller it self. This means that the contents of the following chapters: Access to PLC data and Access to Process data are not valid for the DeviceNet GM alternative.

Regardless of which type of fieldbus interface you use the following model describes how it is interfaced with the PowerMACS system.

Note! In PLC programs generated with the setup wizard from version 7.2.0 the PLC In area starts at address 30000 instead of 3300 and the PLC Out area starts at address 30000 instead of 3000.

On the fieldbus interface there is a memory area that is accessible from the fieldbus master as well as the PowerMACS.

Each fieldbus has its own characteristics regarding the amount of data it could transfer over the bus. The data on the bus is divided in cyclic and acyclic transfer.

Cyclic transfer occurs on a regular basis between the master and the PowerMACS 4000 fieldbus. Cyclic data is typically PLC data.

Acyclic transfer occurs only on a request from the master to the PowerMACS 4000 fieldbus. Acyclic data is typically cycle data, traces etc.

Peripheral Devices

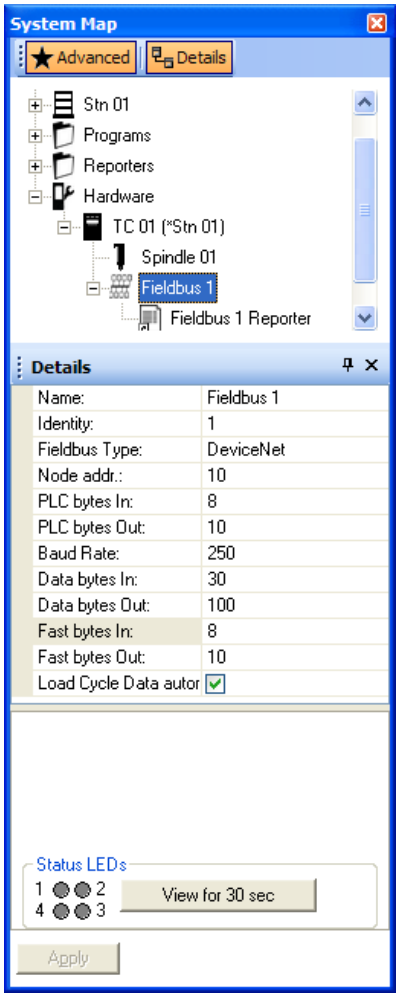
A summary of the features offered by the different fieldbus modules are presented in the table below. All numbers are in bytes.

The maximum available size of the IN and OUT areas varies with the type of fieldbus as listed in the comparison table below, all numbers are in bytes:

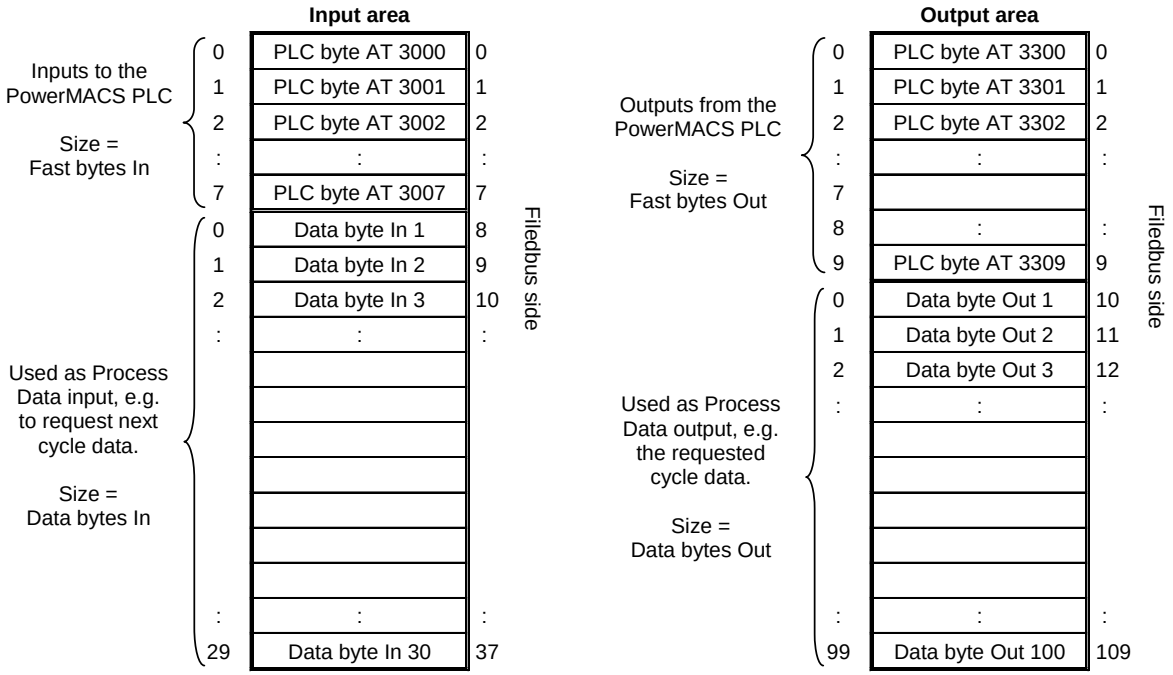
Item	DeviceNet	Profibus DP-V1	EtherNet/IP	Modbus/TCP	ProfiNet IO
Max. Cyclic Data (Read + Write)	512	152(400)[a]	504	512[c]	512[d]
Min. Cyclic Data (Read + Write)	0	1	0	0	0
Max. Write Cyclic Data	256	152(244)[a]	256	256[c]	256[d]
Min. Write Cyclic Data	0	0	0	0	0
Max. Read Cyclic Data	256	152(244)[a]	256	256[c]	256[d]
Min. Read Cyclic Data	0	0	0	0	0
Max. Acyclic Data (Read + Write)	7680	8192[b]	8192	8192	8192
Min. Acyclic Data (Read + Write)	0	0	0	0	0
Max. Write Acyclic Data	7680	8192[b]	8192	8192	8192
Min. Write Acyclic Data	0	0	0	0	0
Max. Read Acyclic Data	7680	8192[b]	8192	8192	8192
Min. Read Acyclic Data	0	0	0	0	0

a. The numbers in parenthesis are valid if the checkbox “Extended Mode” is checked.
b. Only available for DP-V1 protocol.
c. This has to be a multiple of 2 bytes.
d. This has to be a multiple of 8 bytes.

Because the size of the areas is limited you must specify the size of each area individually using the parameters **PLC bytes In**, **PLC bytes Out**, **Data bytes In**, **Data bytes Out, Fast bytes In** and **Fast bytes Out** of the fieldbus device.



The below picture shows how the input and output areas are divided using the parameters:



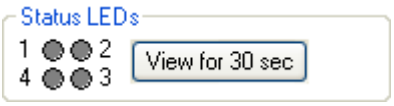
Note! You should not configure the respective input and output areas bigger then what really needed. The reason for this is that all bytes you set up here will be transferred over the fieldbus, regardless if they are used by the application or not.

If you want to transfer Process data over the fieldbus then you must create a **Fieldbus**-Reporter. How to do this, and connect it to your fieldbus device, is described in the New reporter chapter. How to make the Reporter format the result as you wish is described in chapter: Edit reporter.

Indication of the fieldbus status

When connected to a target system you can study the current status of the module using the

System Map form. Select the Fieldbus device, at the bottom of the **Details** window **Status Leds** are shown that indicate the current status of the node.

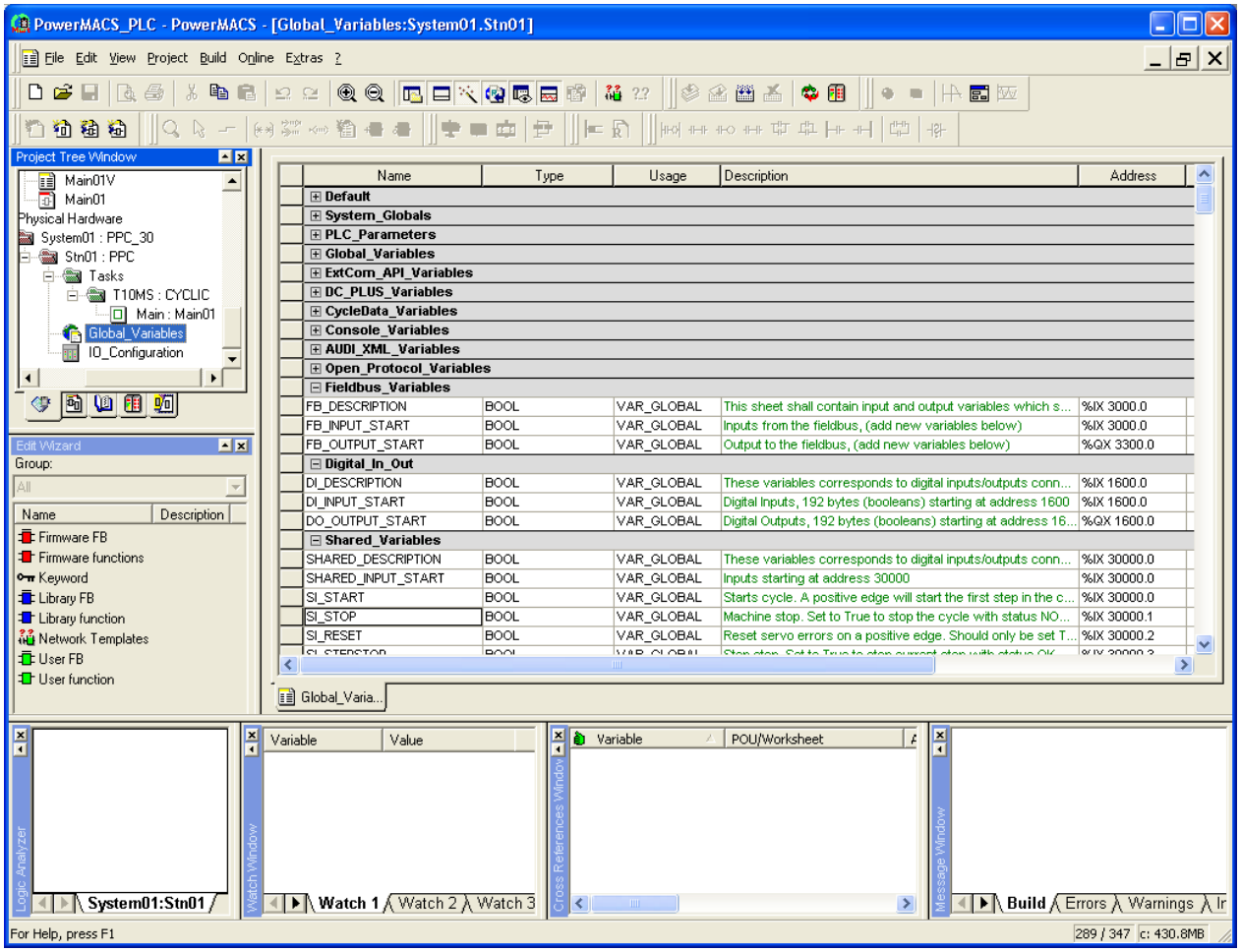


Press the button **View for 30 sec** to start updating the LED's. These will then show the same information as the LED's mounted on the AnyBus board.

The function of the respective LED is specific to the fieldbus type being used. See chapter: Fieldbus specific Information for a description.

8.12.1 Access to PLC data

For ToolsTalk to be able to move around variables in the PLC, Fieldbus variables should be placed in a group called **Fieldbus_Variables** and shared variables in a group called **Shared_Variables**. Placing variables in other groups will result in them not being possible to map for a fieldbus in ToolsTalk. Shared variables must also carry the prefixes “SI_” or “SO_” in order to be mapped by the fieldbus form.


PowerMACS_PLC - PowerMACS - [Global_Variables:System01.Stn01]
File Edit View Project Build Online Extras ?
Project Tree Window
Main01V
Main01
Physical Hardware
System01 : PPC_30
Stn01 : PPC
Tasks
T10MS : CYCLIC
Main : Main01
Global_Variables
IO_Configuration
Edit Wizard
Group:
All
Name
Description
Firmware FB
Firmware functions
Keyword
Library FB
Library function
Network Templates
User FB
User function
Name
Type
Usage
Description
Address
Default
System_Globals
PLC_Parameters
Global_Variables
ExtCom_API_Variables
DC_PLUS_Variables
CycleData_Variables
Console_Variables
AUDI_XML_Variables
Open_Protocol_Variables
Fieldbus_Variables
FB_DESCRIPTION BOOL VAR_GLOBAL This sheet shall contain input and output variables which s... %IX 3000.0
FB_INPUT_START BOOL VAR_GLOBAL Inputs from the fieldbus, (add new variables below) %IX 3000.0
FB_OUTPUT_START BOOL VAR_GLOBAL Output to the fieldbus, (add new variables below) %QX 3300.0
Digital_In_Out
DI_DESCRIPTION BOOL VAR_GLOBAL These variables corresponds to digital inputs/outputs conn... %IX 1600.0
DI_INPUT_START BOOL VAR_GLOBAL Digital Inputs, 192 bytes (booleans) starting at address 1600 %IX 1600.0
DO_OUTPUT_START BOOL VAR_GLOBAL Digital Outputs, 192 bytes (booleans) starting at address 16... %QX 1600.0
Shared_Variables
SHARED_DESCRIPTION BOOL VAR_GLOBAL These variables corresponds to digital inputs/outputs conn... %IX 30000.0
SHARED_INPUT_START BOOL VAR_GLOBAL Inputs starting at address 30000 %IX 30000.0
SI_START BOOL VAR_GLOBAL Starts cycle. A positive edge will start the first step in the c... %IX 30000.0
SI_STOP BOOL VAR_GLOBAL Machine stop. Set to True to stop the cycle with status NO... %IX 30000.1
SI_RESET BOOL VAR_GLOBAL Reset servo errors on a positive edge. Should only be set T... %IX 30000.2
Global_Varia...
Logic Analyzer
System01:Stn01
Watch Window
Variable
Value
Watch 1
Watch 2
Watch 3
Cross References Window
Variable
POU/Worksheet
Message Window
Build
Errors
Warnings
For Help, press F1
289 / 347
c: 430.8MB

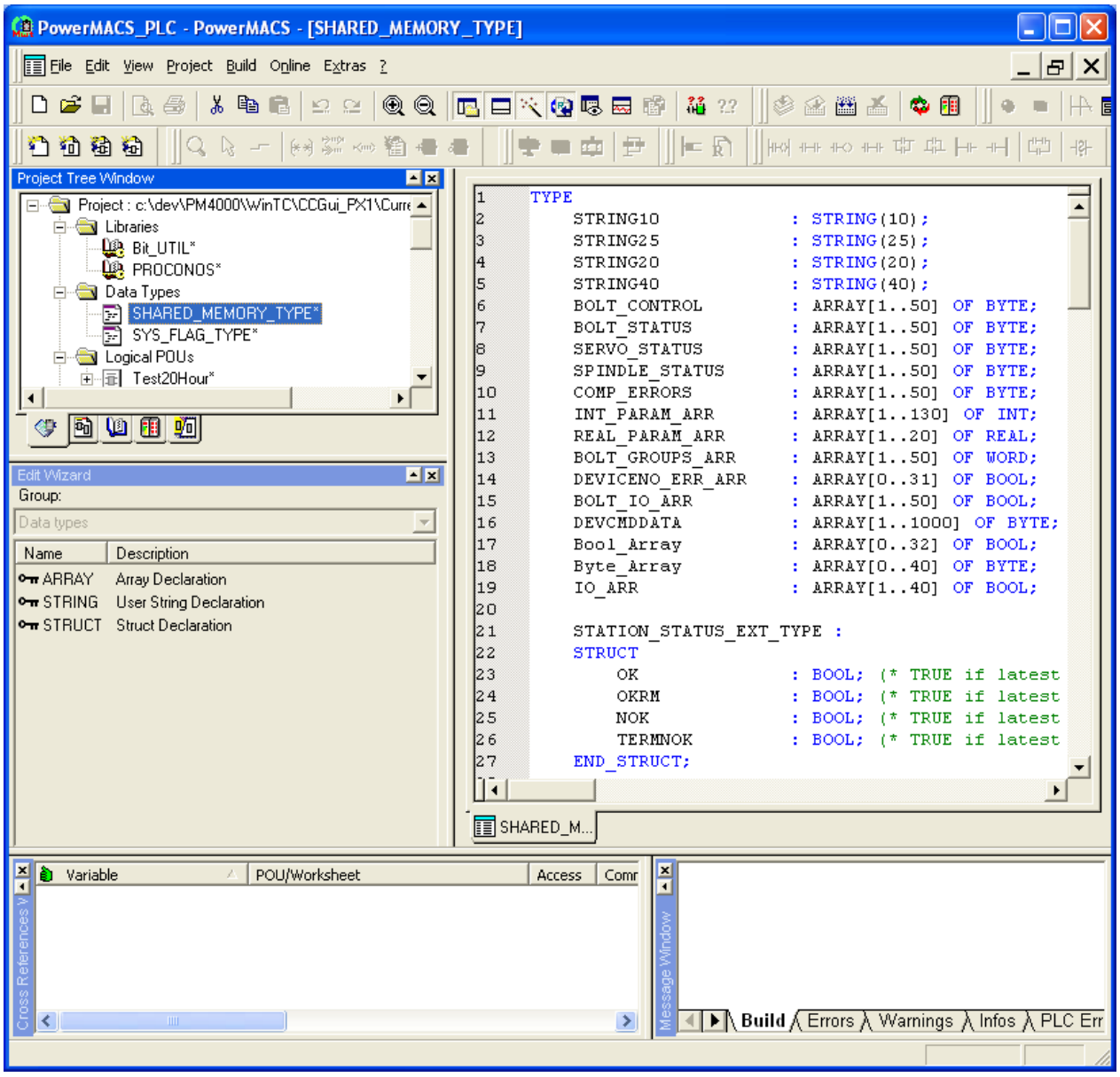
8.12.1.1 PLC Areas

- **Fieldbus input signals**
  Address: 3000-3299
  Can in ToolsTalk be mapped for fieldbus when having prefix “SI_”.
- **Fieldbus output signals**
  Address: 3300-3599
  Can in ToolsTalk be mapped for fieldbus and I/O when having prefix “SO_”.
- **Digital input signals**
  Address: 1600- 1791
  Can in ToolsTalk be mapped only for I/O.
- **Digital output signals**
  Address: 1600- 1791
  Can in ToolsTalk be mapped only for I/O.
- **Shared input signals**
  Address: 30000-30999
  Can in ToolsTalk be mapped for for fieldbus and I/O when having prefix “SI_”.
- **Shared output signals**
  Address: 30000-30999
  Can in ToolsTalk be mapped for fieldbus and I/O when having prefix “SO_”.

NOTE: Only boolean signals can be mapped by I/O.

8.12.1.2 Complex data types

PM4000 has support for user-defined data types and arrays defined in SHARED_MEMORY_TYPE.

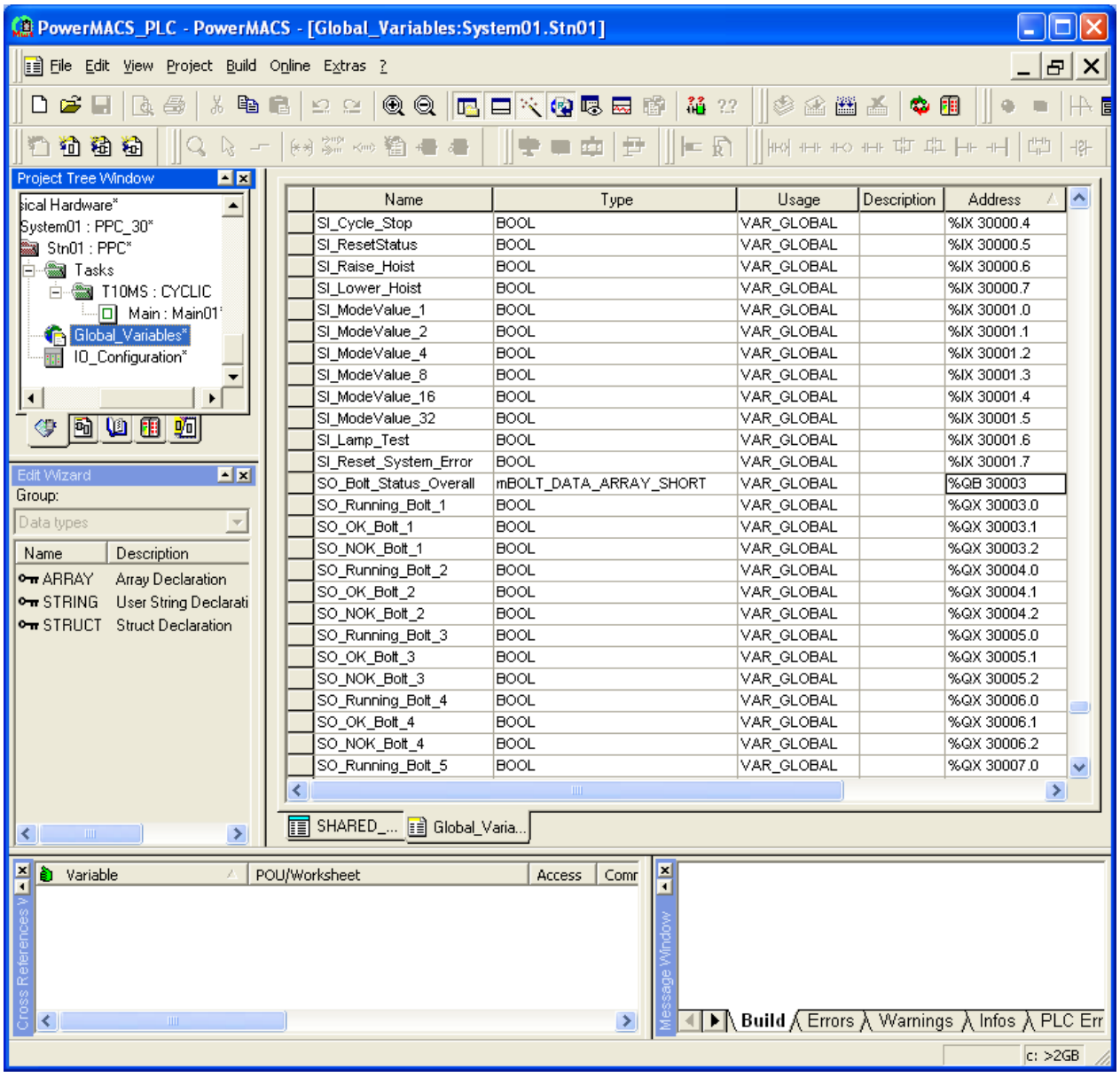


Both a complex data type and each element in the complex data type can be declared in the Global_Variables sheet. PM4000 calculates the size of the complex type to find out how much of the area is occupied by that variable.

When selecting a variable for mapping to a fieldbus in TTPM, all variables declared on addresses occcupied by the complex variable are hidden from the user. When a complex variable is moved in TTPM, all variables declared on addresses occcupied by the complex variable are moved as well.

When mapping a I/O device in TTPM, only boolean variables are available to choose from. Complex data types are hidden from the user.

Example

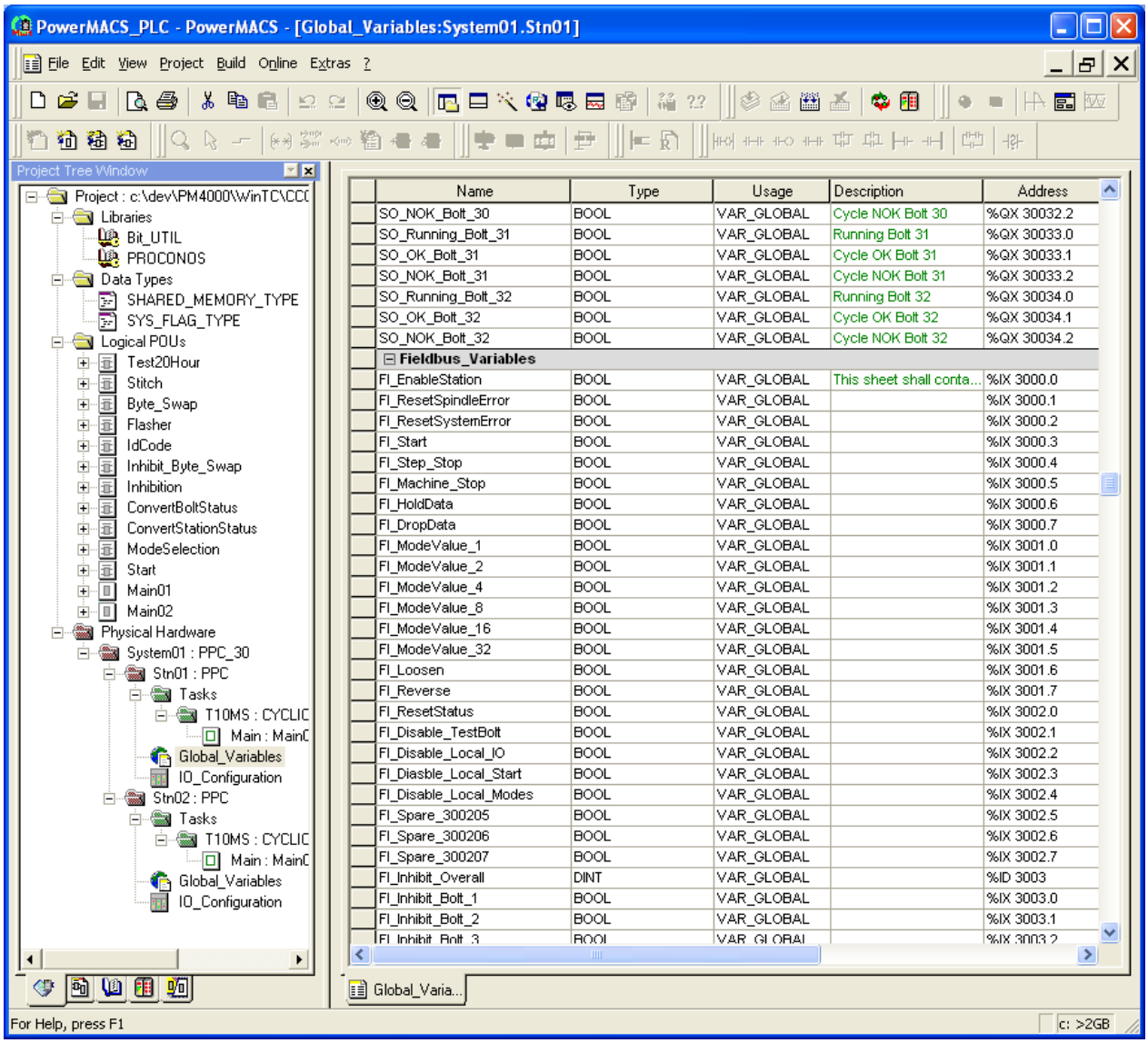


The variable SO_Bolt_Status_Overall has a complex data type and is declared on address 30003. For this example, the size of mBOLT_DATA_ARRAY_SHORT is 50 bytes. The following variables represent the elements within mBOLT_DATA_ARRAY_SHORT and are therefore also declared on addresses starting with 30003. These variables (SO_Running_Bolt_1, SO_OK_Bolt_1, SO_NOK_Bolt_1, SO_Running_Bolt_2 etc.) are not available when mapping a fieldbus in TTPM. They are instead used when mapping I/O devices, while SO_Bolt_Status_Overall is not.

If SO_Bolt_Status_Overall is mapped from the fieldbus form in TTPM, all variables between 30003 and 30052 are moved to the fieldbus area.

8.12.1.3 Set up shared variables

A number of signals are predefined in the PLC, but signals can be added, delete, or get a changed functionality. Signals under the group **Shared_Variables** in the Global variables sheet in the PLC can be shared between I/O and fieldbus. Signals under the group **Fieldbus_Variables** represent the fieldbus mapping.



When converting a setup older than version 7.5.0, the PLC will not have the same signals defined. Access to the PLC part of the fieldbus data area from the PowerMACS PLC is configured in two steps. First you must configure how the respective area should be used by defining variables. Secondly must configure

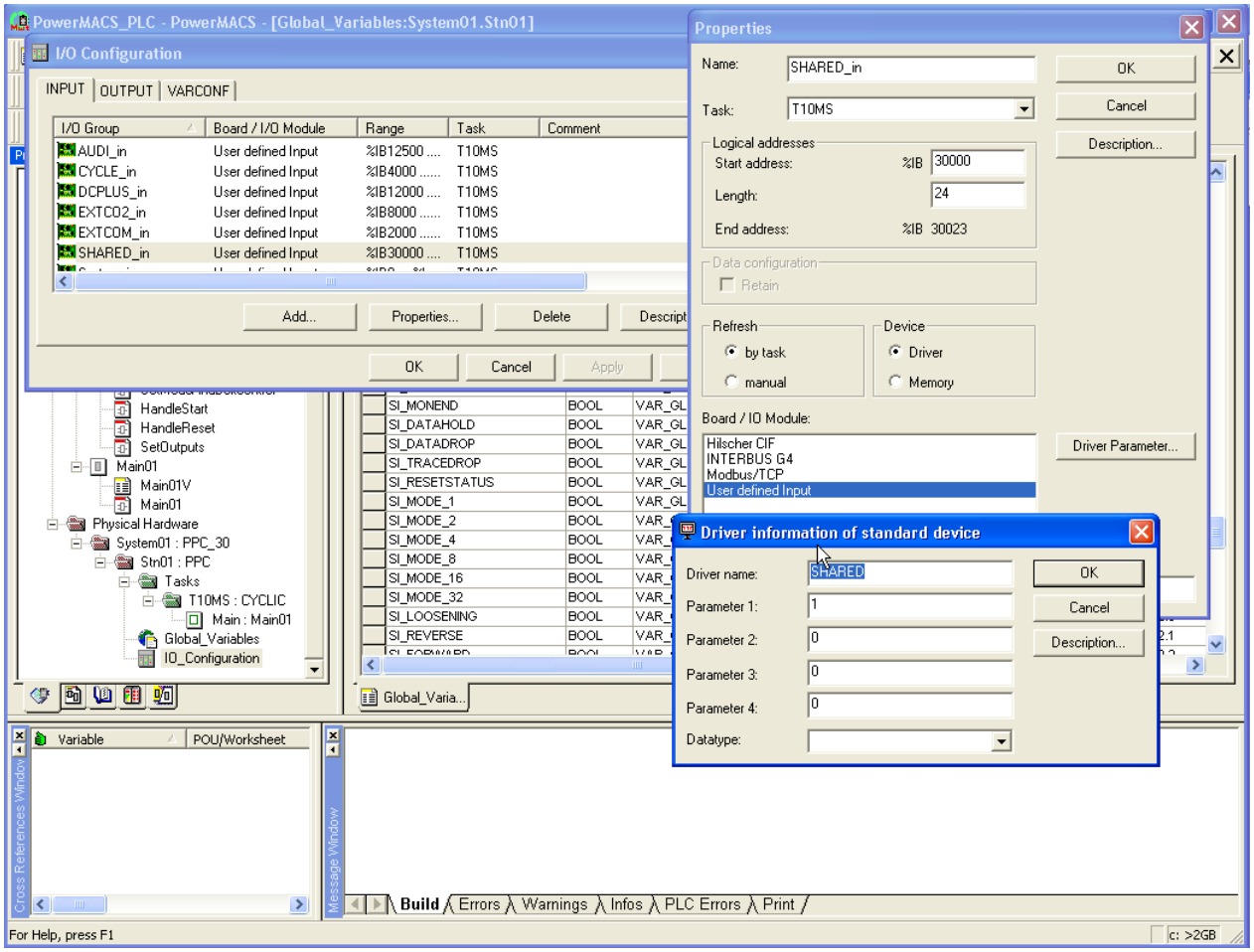
the SHARED driver to correctly map the total sizes of the respective input and output data areas defined in the first step.

The addresses for inputs and outputs starts at 30000 and are logical only. Signals added here must also have the prefixes “SI_” or “SO_” to be able to map to fieldbus and I/O devices.

Note! All numerical values are written and read as Big Endian values, commonly referred to as Motorola format.

When the variables are declared you are free to refer them from any POU (Program Organization Unit) you have, or will, declare in the PLC project.

After mapping the input and outputs variables the next step is to configure the SHARED driver to copy the input and output data from the fieldbus to the PLC and vice versa. This is done using the I/O Configuration dialogue which is invoked by double-clicking on the icon for IO_Configuration worksheet of station that has the fieldbus interface mounted.

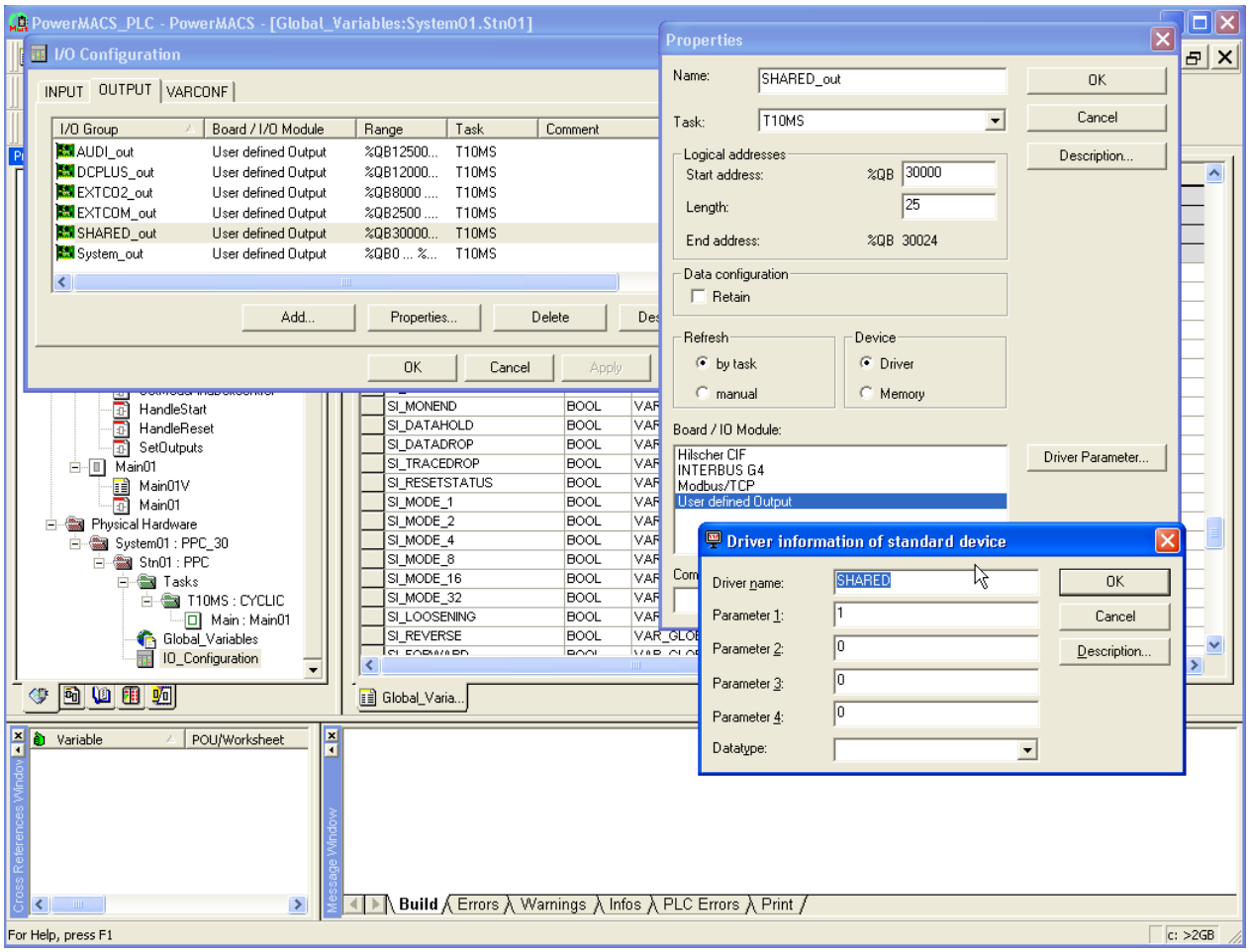


In the I/O Configuration dialogue first select the INPUT tab, select the I/O Group SHARED_in and press the Properties button. The group name has no functionality, therefore the group can be named anything.

Change the value of Logical addresses – Length so that it corresponds to the size of your fieldbus data declared as inputs, in this case 24.

Then select User defined Input and click the Driver Parameter-button to set up the driver information. Name the driver SHARED.

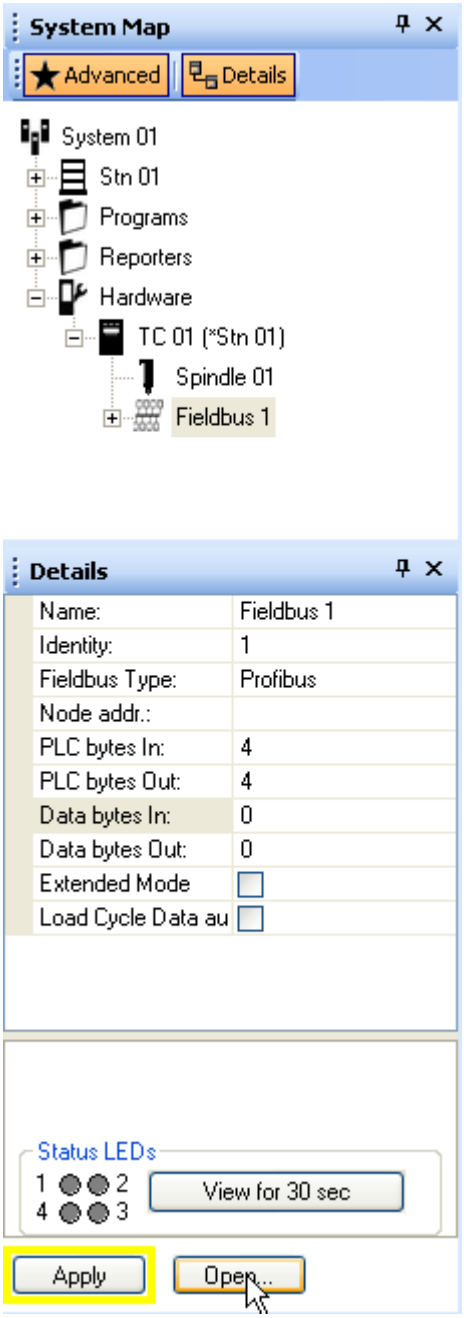
Then select the OUTPUT tab in the I/O Configuration dialogue, select the I/O Group SHARED_out and press the Properties button. The group name has no functionality, therefore the group can be named anything.



Set up address, length and driver parameter the same way as for the input driver. For both input and output driver, the address should be 30000, driver name should be SHARED. The length can differ based on how much data is needed for input and output variables.

8.12.1.4 Fieldbus form

The fieldbus form is reached by pressing the "Open" button in the system map, or by double-clicking the fieldbus directly in the map.

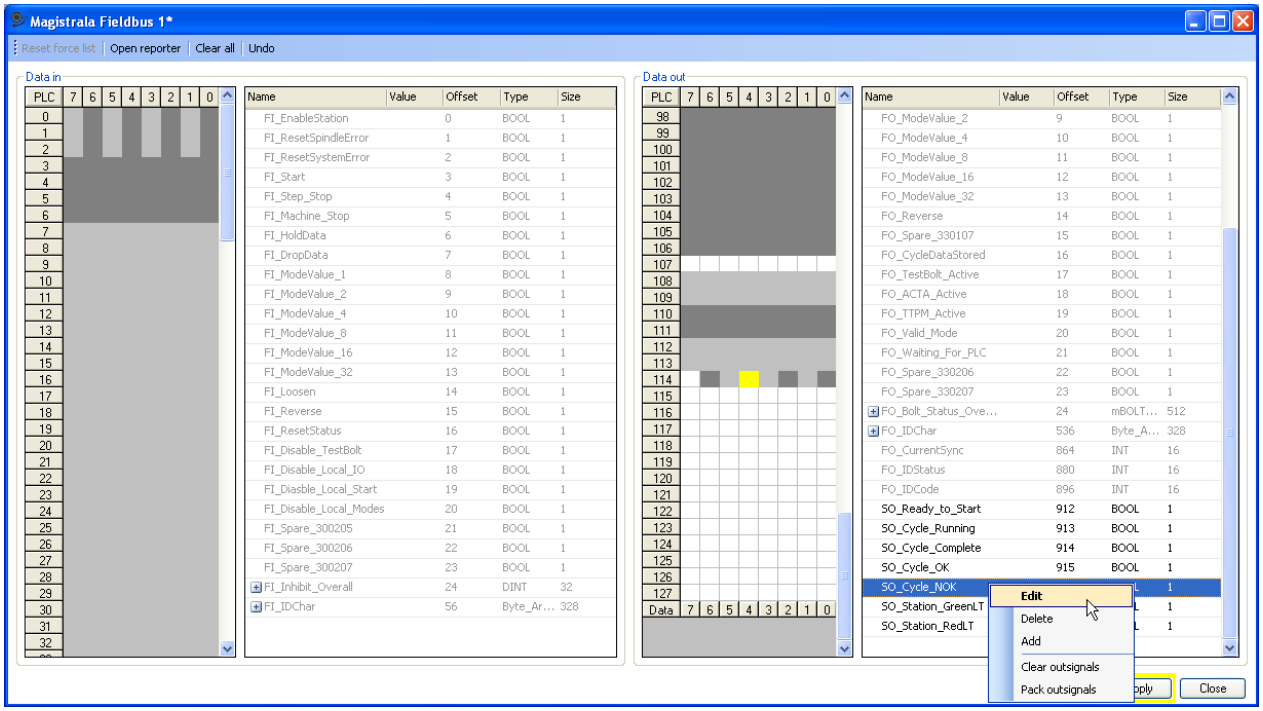


The form shows a listview and a memory grid each for data in and data out. The form represent the signals in the **Fieldbus_Variables** group in the PLC.

When clicking on a cell in the grid, the mapped signal is highlighted in the listview and the whole signal is highlighted in the grid. When clicking on a signal in the listview, the signal is highlighted in the grid.

Mapping of PLC signals can only be done when not connected to a TC. Monitoring and forcing of PLC signals can only be done when connected.

Data bytes can not be set up in this form. A link to the reporter form is available on the command bar. Only monitoring is available for the data bytes.



Mapping of signals

When not connected to a TC, PLC signals is added either by dragging them from the dynamic tool, which will have one list of the available insignals and one list of the available outsignals, or by right-clicking one of the listviews.

By right-clicking a signal in any of the lists, the user can choose to edit the offset, delete signal, clear all signals in that list, or add new signals. The alternative Pack signals in the context menu will fill holes in the map by moving all signals towards offset 0. Signals can also be dragged and dropped if the user wants to change the order of the signals.

The "data" part of the grids will show nothing during the mapping phase.

Only shared signals can be mapped in the fieldbus form. A shared signal is defined by having its name start with “SI_” or “SO_”.

Complex data types can be mapped even if their size exceeds the declared fieldbus size. Regular data types will not be added if their size exceeds the declared fieldbus size.

When pressing the **Apply** button, the mapped signal in the fieldbus are being moved to the defined address in the fieldbus area in the PLC.

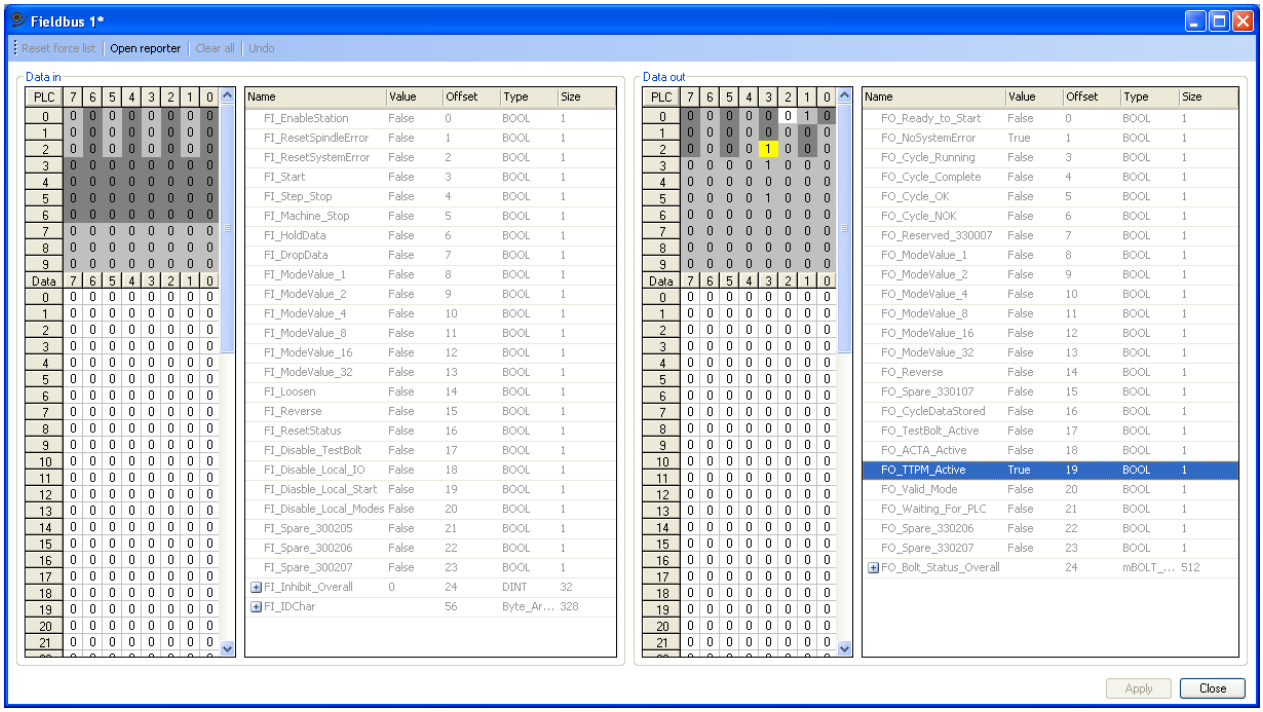
A signal being deleted in the fieldbus form is moved back to the first free address in the shared area in the PLC when pressing **Apply**.

Monitoring

When connected to a TC, fieldbus data is monitored in the form. A snapshot of the fieldbus data is taken once a second and sent to TTPM from the TC. The monitored values of the PLC signals are based on the fieldbus data and not taken directly from the PLC. This means that if the user connects to a TC without having pressed Apply in the fieldbus form, the signal values in the fieldbus form might not be the same as in the PLC.

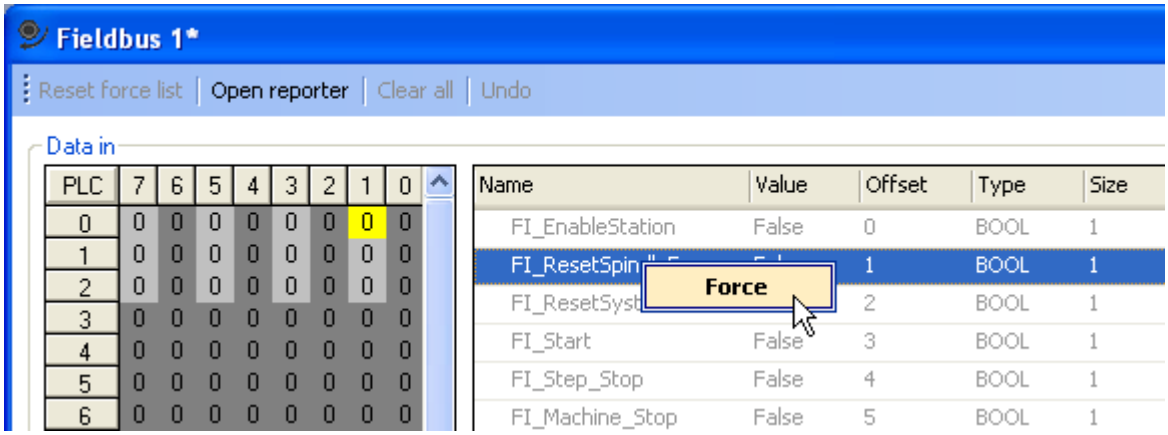
Values for signals and variables will be shown in the listview. The "value" column in the listview is empty when not connected.

Since the refreshing of values in this form is slower than then PLC, very quick data changes in the PLC can occur without being noticed in this form.



Force values

When connected to a TC, a signal can be forced from the fieldbus form by right-clicking the signal and selecting Force in the context menu. The values of forced signals are marked by a star in the form. The data is forced onto the fieldbus data map and not directly into the PLC. This means that if the user connects to a TC without having pressed Apply in the fieldbus form, the forced values in the fieldbus form might not force the same signal in the PLC.



The force list can be reset from the command bar. Values can only be forced when connected to a TC and when the fieldbus form is open. On disconnection from the TC, the TC will reset the force list automatically.

Only the PLC bytes can be forced.

8.12.2 Access to Process data

Accesses to the Process data, i.e. Cycle data, events, trace and setup, by the fieldbus master are done by read and write accesses to the Process data areas.

Because Process data normally is bigger than would fit into the areas available they are read by use of a special handshake sequence. This is described in chapter: Process data.

Also, since the same area (address) is used for accesses to all kinds of process data the input and output areas are prefixed with a command and a status word respectively. These are used to specify what to do, and with what kind of data. The Process data areas are therefore used as follows:

Byte	Process data IN area
0	CMD (Most Significant Byte)
1	CMD (Least Significant Byte)
2	
3	Data
:	TO
	PowerMACS
:	
Data bytes In – 1	

Byte	Process data OUT area
0	STS (Most Significant Byte)
1	STS (Least Significant Byte)
2	
:	Data
	FROM
	PowerMACS
:	
Data bytes Out – 1	

The first word in the IN area is used as Command (CMD). The fieldbus master writes commands in this to request PowerMACS perform a task.

The first word in the OUT area is used as Status (STS) in which PowerMACS writes information to indicate the result of the command issued.

The CMD word is laid out as follows:

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	R	W	F	S	X	d2	d1	d0	-	-	-	-	-	-	-	-

The respective bits have the following meaning:

Name	**Meaning**	**Usage**
R	Read	Set one (1) to read the process data selected with DataType.
W	Write	Set one (1) to write the process data selected with DataType.
F	Flush	Set one (1) to flush the data queue selected with DataType. Flushing will delete all items in the specified process data queue. However, a data item currently being read will not be deleted. It can safely be read also after that this command has been issued.
S	Skip	Set one (1) to skip reading of the current data item in the data queue selected with DataType. Next reading will return next data item (or the first part of it) in the queue.
X	Extended format	This bit is normally 0. If this bit is set then which process data queue to read is **not** specified using the d2..d0. Instead the queue is specified using an address following the CMD byte. See Using the extended format for a detailed description. This can be used for to access more data to the PLC then the particular fieldbus allows.
d2..d0	DataType	Specifies which process data queue the command is issued for. DataType = $4^{d2} + 2^{d1} + 1^{d0}$ d. Allowed values of DataType are: • 1: Cycle data queue • 2: Event queue • 3: Trace queue • 4: Setup

The STS word is used as follows:

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Name	R	W	F	S	X	d2	d1	d0	s7	s6	s5	s4	s3	s2	s1	s0

Bit 15 to 8 just mirrors the corresponding bits in the CMD word.

The bits named s7 to s0 represents the status code that PowerMACS returns after executing the command. They should be interpreted as a signed 8 bit value where a value greater or equal to zero indicates a positive result, and negative values a failure. Currently the following status codes are defined:

Status code	**Description**
0	Success
-1	Illegal process data specified
-2	Illegal size of the process data area
-3	Device is not properly connected
-4	Error in sequence number
-5	Received setup is to big
-15	Unknown error
-16	Size of Data bytes In to small for the requested service
-17	Size of Data bytes Out to small for the requested service
-18	The specified Process Data type cannot be flushed/skipped
-19	Ambiguous Command (more than one of the R, W, F & S bits are set)
-20	Invalid Extended format
-21	Invalid Address
-22	Fieldbus read error
-23	Fieldbus write error

When the Fieldbus master wants to read or write anything via the PowerMACS process data interface it has to follow the following basic rules:

1. Read the STS word to ensure that it is zero.
2. If not zero then write zero to the CMD word to acknowledge that the previous data/result has been read.
3. Wait for the STS word to become zero. This is PowerMACS way to indicate that it is ready to receive a new command.
4. Write the wanted command to the CMD word.
5. Wait for the high byte (bit 15...8) of the STS word to become equal to the written in the CMD word. This is PowerMACS way to acknowledge that the command is received and executed.
6. Check that the low byte (bit 7...0) of the STS word is zero. This indicates that the command was executed successfully. If reading, the output data is now valid.
7. When finished, with reading of data and/or the STS word, write zero to the CMD word to acknowledge that the result is read.

Example on how to read process data:

To read Process data the fieldbus master should use the following handshake sequence:

	Fieldbus master	**PowerMACS**
1	Sets CMD = R + DataType	
2		Loads data in the OUT area Sets STS = R + DataType + Status
3	Reads data from the OUT area Sets CMD = 0	
4		Sets STS = 0
5	Waits until STS = 0	

Example on how to write process data:

To write Process data the fieldbus master should use the following handshake sequence:

	Fieldbus master	**PowerMACS**
1	Writes data in the IN area Sets CMD = W + DataType	
2		Reads data from the IN area Sets STS = W + DataType + Status
3	Sets CMD = 0	
4		Sets STS = 0
5	Waits until STS = 0	

Whenever one of these sequences is executed a new package of data is transferred to, or from, the PowerMACS system.

Since most types of Process data are too big to fit into the available area it is split up into packages. Each access will transfer a new package. The data part of the IN and the OUT areas are therefore further dived into several fields in order to handle transferring sequence of packages.

See chapter: Process data for a detailed description of this. There is also described how the respective Process data is formatted.

Normally no data is written to the Process Data out area unless the client first has requested so. However, by checking the **Load Cycle Data automatically** check box for the fieldbus device (see System) the PowerMACS system will automatically load the first package of a cycle data when the cycle is finished.

The data will only be written if the Process Data out area is free for PowerMACS to write to. The area is considered free as long as the fieldbus master do not already have a read or write transaction ongoing (i.e. it have issued a command but not yet written zero (0) to the CMD word to indicate that it has read the response). If a Cycle Data is generated when the output area is not free then it will be lost. It will **not** be written automatically when the area becomes free again.

An automatically written cycle data is formatted exactly the same way as if the fieldbus master had used the CMD to request it. That is, in addition to the result data, the STS, SEQ, and LEN fields are also properly assigned.

After reading automatically loaded data the fieldbus master **do not** need to indicate that he is done reading by writing zero (0) to the CMD word.

All normal Process Data functions over the fieldbus are supported in parallel. This means that the fieldbus master may request Events, Traces as well as Cycle Data's (from the FIFO queue) if so wanted. However, the master must be aware of that doing so may interfere with the automatic loading of Cycle Data. Typically the master should issue such commands only in-between cycles and after the last generated Cycle Data have been read.

Note 1: It is only the first package of a cycle data that is written automatically to the board. This means that if the cycle data is bigger then the Process data output area, i.e. Data bytes Out, then the client must request the remaining packages by using normal read request. See chapter: Process data for a description detailed description of how a cycle data is divided in packages.

Note 2: When using **Load Cycle Data automatically** the fieldbus master must ensure that it is done reading data before starting a new cycle.

8.12.2.1 Using the extended format

The Extended format gives you an alternative method of specifying the targeted type of process data.

When using this method you specify what to read or write to using an address instead of the bits d2..d0 in the command word. The address is the same as used for the Serial Communication (see its sub chapter: Accesses from External communication device).

An advantage of using this method is that you in addition of accessing the traditional process data, i.e. cycle data, trace, and events, also can read and write to the PLC input/output area normally used for the External communication devices and the API.

The address is specified in the first word of the Process data, i.e. the word following next after the command word. I.e. Process data should be laid out as below:

Process Data

0	Target Addr (MSB)
1	Target Addr (LSB)
2	
3	Message
	Data
:	(m bytes)
m + 1	

The layout of Message Data depends on whether the targeted data is of type "Process Data" (i.e. Cycle Data, Event Data, Trace Data and Setup Data) or "PLC Data".

Message Data to/from the PLC

0	LEN (MSB)
1	LEN (LSB)
2	Data byte 1
3	Data byte 2
:	
n + 1	Data byte n

LEN

Message Data as Process Data

0	SEQ (MSB)
1	SEQ (LSB)
2	LEN (MSB)
3	LEN (LSB)
4	Data byte 1
5	Data byte 2
:	
n + 3	Data byte n

LEN

The format of Process Data is the very same as used the External communication devices, but the format of PLC Data is slightly different. For the fieldbus device the first word of the Message Data specifies the length of data to be read or written. If writing, this is then followed with as many bytes of PLC data that is specified.

Also, note that if the extended format is used for the command then the target address of the command will be included in the response as well.

See chapter: Process data for a description of how the Process Data is laid out and how SEQ and LEN is used.

Example: To write the ten characters ID code "ABCDEFGHIJ" to the PLC input starting at address 2010 the following should be written to the fieldbus process data input:

0	0x48	CMD (MSB)
1	0x00	CMD (LSB)
2	0x00	Target Addr. (MSB)
3	0x0A	Target Addr. (LSB)
4	0x00	LEN (MSB)
5	0x0A	LEN (LSB)
6	0x41	A
7	0x42	B
8	0x43	C
9	0x44	D
10	0x45	E
11	0x46	F
12	0x47	G
13	0x48	H
14	0x49	I
15	0x4A	J

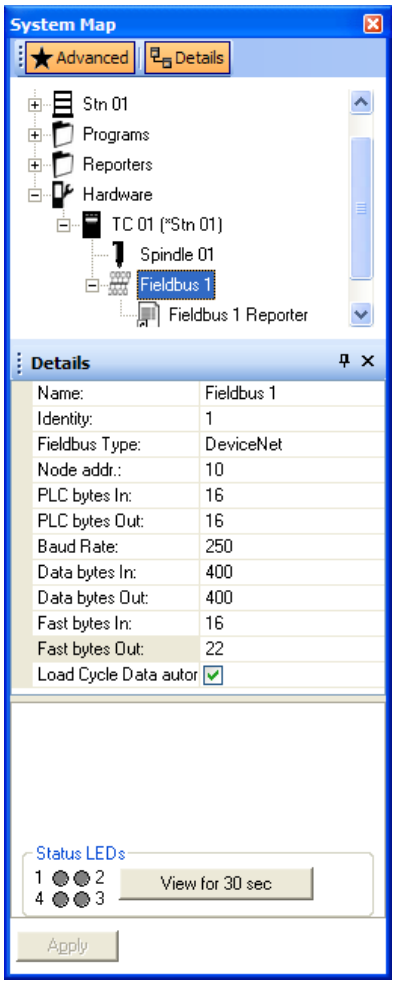
8.12.3 Fieldbus specific Information

Most field buses are handled the same way by the PowerMACS system but some of them have a number of extra parameters. These are described in the following sub chapters.

8.12.3.1 DeviceNet

The Anybus-CC DeviceNet communication module provides instant DeviceNet Real Time connectivity via the host interface and supports the following features:

- DeviceNet communication, with baud rates 125, 250 and 500kbps
- Up to 256 bytes of Real Time I/O
- Up to 7060 bytes of acyclic data such as cycle data, trace etc.
- Generic EDS-file.



Use **Name** to specify the name of the device. This name should be referred to when you create a reporter for the fieldbus using the **New reporter** form.

Fieldbus Type should be set to DeviceNet.

Use **Node addr.** field to set the node address. Node address can be set to [0..63]. Use the **Baud Rate** to set the baud rate to 125, 250 or 500 kBit/s.

See chapter: **Fieldbus Interface** for a description on **PLC bytes In**, **PLC bytes Out**, **Data bytes In**, and **Data bytes Out.**

The parameter **Fast bytes In** defines the size (in bytes) of the input data that is transferred as cyclic, or real time, IO data on the DeviceNet network. Max value is **PLC bytes In** + **Data bytes In.** Leaving the parameter "blank" will make all input data transferred as IO data.

Fast bytes Out defines the size (in bytes) of the output data that is transferred as cyclic IO data on the DeviceNet network. Max value is **PLC bytes Out** + **Data bytes Out**. Leaving the parameter “blank" will make all output data transferred as IO data.

Note: The max size of **Fast byte In/ Out** is depended on the fieldbus type and could be found in the table in chapter **Fieldbus Interface**.

The "**fast**" areas always start at offset zero (0) of the respective input or output area. The remaining part of the respective areas is transferred as acyclic data.

The input as well as the output area of the DeviceNet module is divided in data blocks. The maximum size of each block is 64 bytes. For example, if 400 bytes output is configured it will be available as seven

blocks, the first six contains 64 * 6 = 384 bytes and the seventh 16 bytes.

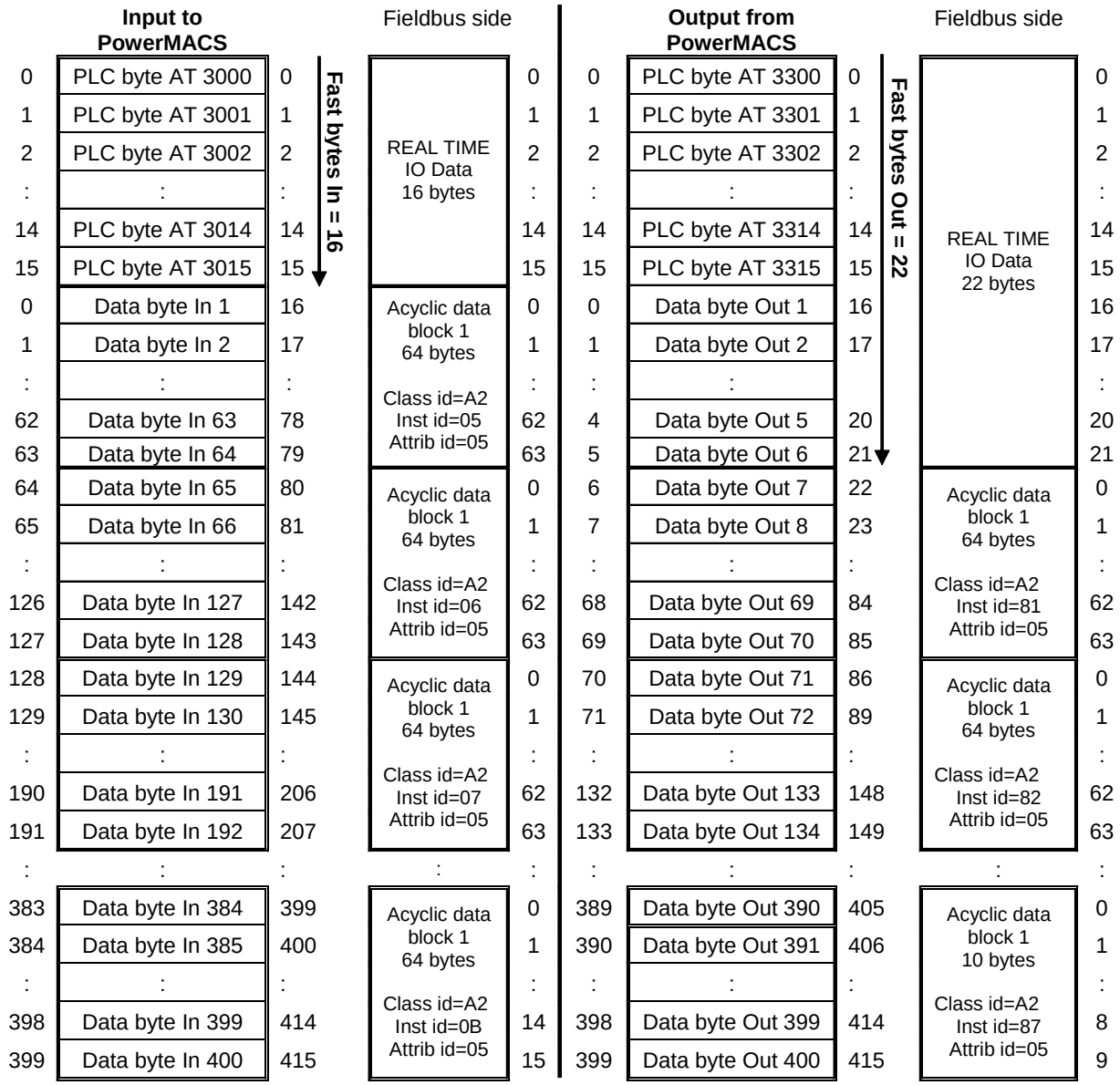
Inputs to PowerMACS (that is, the AnyBus OUT area) are mapped as follows:

Class Id	Acyclic block No (bytes)	Instance Id	Attribute Id
0xA2 - Acyclic Data Input	1 (0 .. 63)	0x05	0x05
	2 (64 .. 127)	0x06	0x05
	3 (128 .. 191)	0x07	0x05
:	:	:	:
	118 (7488 .. 7551)	0x7B	0x05
	119 (7552 .. 7615)	0x7C	0x05
	120 (7616 .. 7679)	0x7D	0x05

Outputs from PowerMACS (that is, the AnyBus IN area) is mapped as follows:

Class Id	Acyclic block No (bytes)	Instance Id	Attribute Id
0xA2 - Acyclic Data Output	1 (0 .. 63)	0x81	0x05
	2 (64 .. 127)	0x82	0x05
	3 (128 .. 191)	0x83	0x05
:	:	:	:
	118 (7488 .. 7551)	0xF7	0x05
	119 (7552 .. 7615)	0xF8	0x05
	120 (7616 .. 7679)	0xF9	0x05

With parameters set as in the above example the input and output areas are mapped as below:

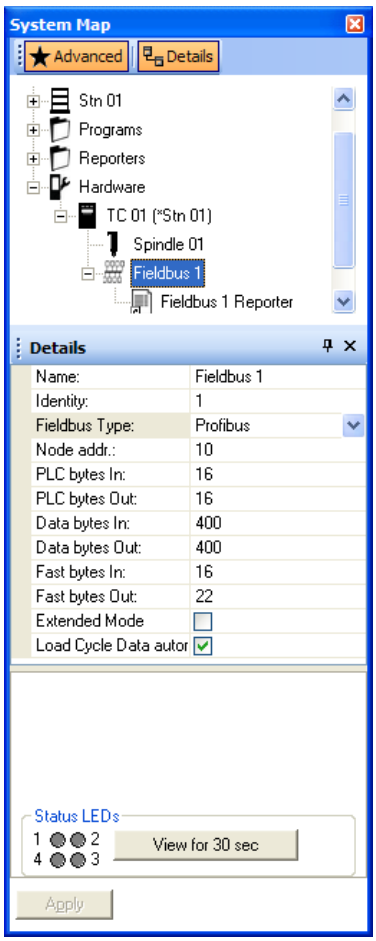


Three LED's on the AnyBus module indicate the status of the DeviceNet. These LED:s can also be displayed from the **System Map** form, see **Indication of the fieldbus node's** for how to do this.

The function of the respective LED is as follows:

LED	Function	Indicates
1	**Network status**	**Steady off** – Not Online / No Power.
		Steady Green – On-line, one or more connections established.
		Steady Green (1 Hz) – On-line, no connections established.
		Steady Red – Critical Link Failure.
		Flashing Red (1 Hz) – One or more connections timed-out.
2	**Module status**	**Steady off** –No Power.
		Steady Green – Operating in normal condition.
		Flashing Green (1 Hz) – Missing or incomplete configuration.
		Steady Red –Major Fault(s).
		Flashing Red (1 Hz) – Minor Faults(s).

8.12.3.2 Profibus DP and DP-V1



The Anybus-CC Profibus communication module provides instant Profibus Real Time connectivity via the host interface and supports the following features:

- Profibus communication, with baud rates 12 Mbps (auto detect by the module)
- Up to 152/244 bytes of Real Time I/O
- Up to 8192 bytes of acyclic data such as cycle data, trace etc.
- Acyclic data can be accessed from the network using DP-V1 read/write services.
- Generic GSD-file.

Use **Name** to specify the name of the device. This name should be referred to when you create a reporter for the fieldbus using the **New reporter** form.

Fieldbus Type should be set to DeviceNet.

Use **Node addr.** field to set the node address. Node address can be set to [0..125].

See chapter: **Fieldbus Interface** for a description on **PLC bytes In**, **PLC bytes Out**, **Data bytes In**, and **Data bytes Out.**

The parameter **Fast bytes In** defines the size (in bytes) of the input data that is transferred as cyclic, or real time, IO data on the ProfiNet IO network. Max value is **PLC bytes In** + **Data bytes In.** Leaving the parameter "blank" will make all input data transfererred as cyclic IO data.

Fast bytes Out defines the size (in bytes) of the output data that is transferred as cyclic IO data on Profibus network. Max value is **PLC bytes Out** + **Data bytes Out**. Leaving the parameter “blank" will make all output data transferred as cyclic IO data.

The "**fast**" areas always start at offset zero (0) of the respective input or output area. The remaining part of the respective areas is transferred as acyclic data.

The input as well as the output area of the Profibus module is divided in data blocks. The maximum size of each block is 64 bytes. For example, if 400 bytes output is configured it will be available as seven blocks, the first six contains 64 * 6 = 384 bytes and the seventh 16 bytes.

The parameter **Extended Mode** is a checkbox that chooses between the **152** bytes data mode and the **368** bytes data mode. When this checkbox is unchecked the backward compatible **152** bytes data mode is chosen, and when checked the new **368** bytes data mode is chosen.

A GSD-file for the different data modes are included in the installation kit and are named **321-2932-HMS_1811.gsd** for the **152** bytes data mode and **Ext_321-2932-HMS_1811.gsd** for the **368** bytes data mode.

Note! The **368** bytes data mode requires that the AnyBus module has at least software and hardware versions greater than or equal to 2.0. It is likely that Profibus modules already delivered with systems are too old and cannot be used for this mode. If the **368** bytes data mode is used on an old module the following event will be generated ***ANYBUS: TC %d: Error! Module version (%x) does not support Extended Mode***, and the module has to be replaced. When using the **368** bytes data mode there is a limitation of max **244** bytes input or output. ***Do not use extended mode if 152 bytes of fast data is enough for your application***.

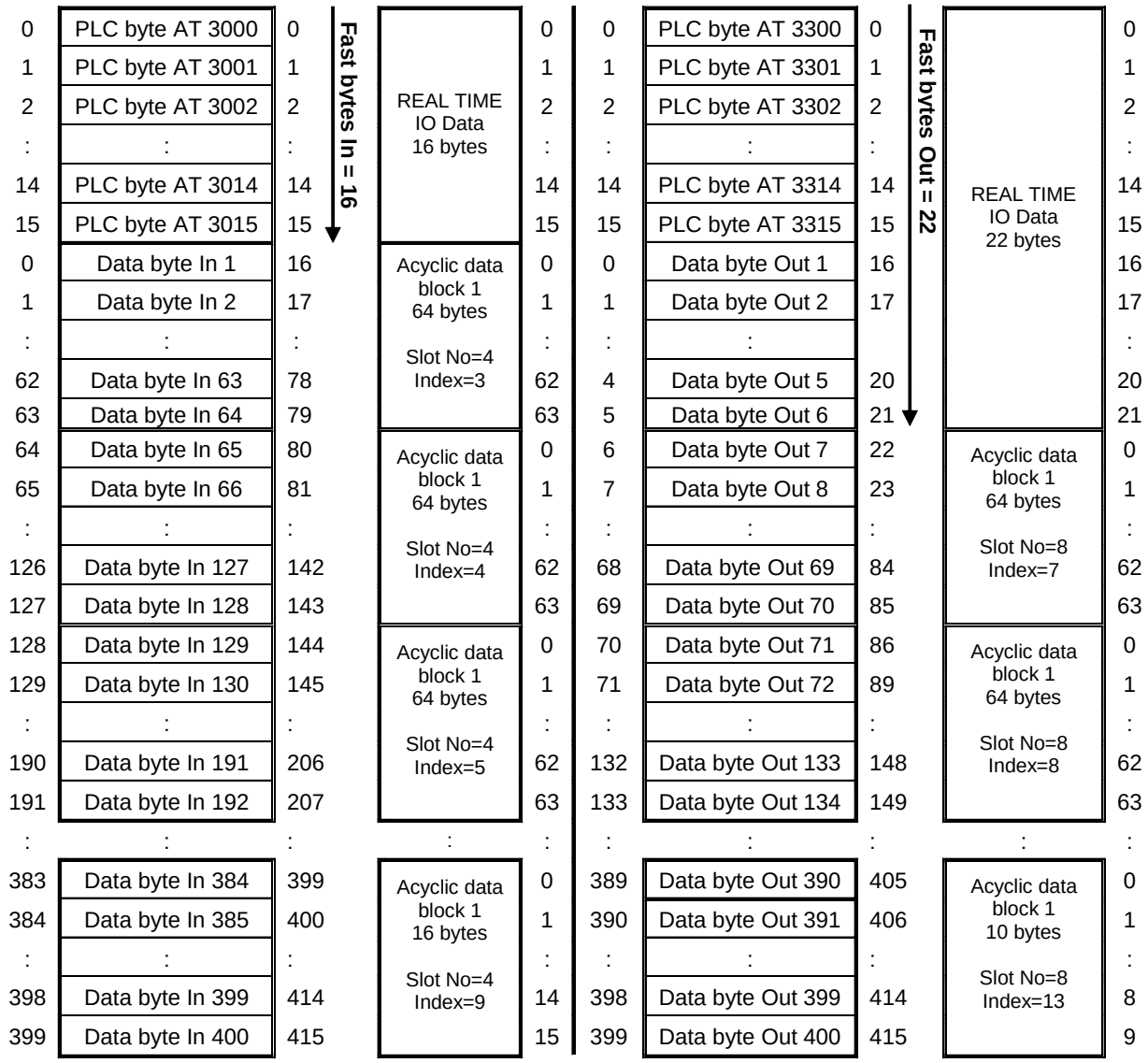
Inputs to PowerMACS (that is, the AnyBus OUT area) are mapped as follows:

Command	Acyclic block No (bytes)	Slot No	Index
DP-V1 / write services	1 (0 .. 63)	4	3
	2 (64 .. 127)	4	4
	3 (128 .. 191)	4	5
:	:	:	:
	126 (8000 .. 8063)	4	128
	127 (8064 .. 8127)	4	129
	128 (8128 .. 8191)	4	130

Outputs from PowerMACS (that is, the AnyBus IN area) is mapped as follows:

Command	Acyclic block No (bytes)	Slot No	Index
DP-V1 / read services	1 (0 .. 63)	8	7
	2 (64 .. 127)	8	8
	3 (128 .. 191)	8	9
:	:	:	:
	126 (8000 .. 8063)	8	132
	127 (8064 .. 8127)	8	133
	128 (8128 .. 8191)	8	134

With parameters set as in the above example the input and output areas are mapped as below:



Three LED's on the AnyBus module indicate the status of the Profibus. These LED:s can also be displayed from the **System Map** form, see **Indication of the fieldbus node's** for how to do this.

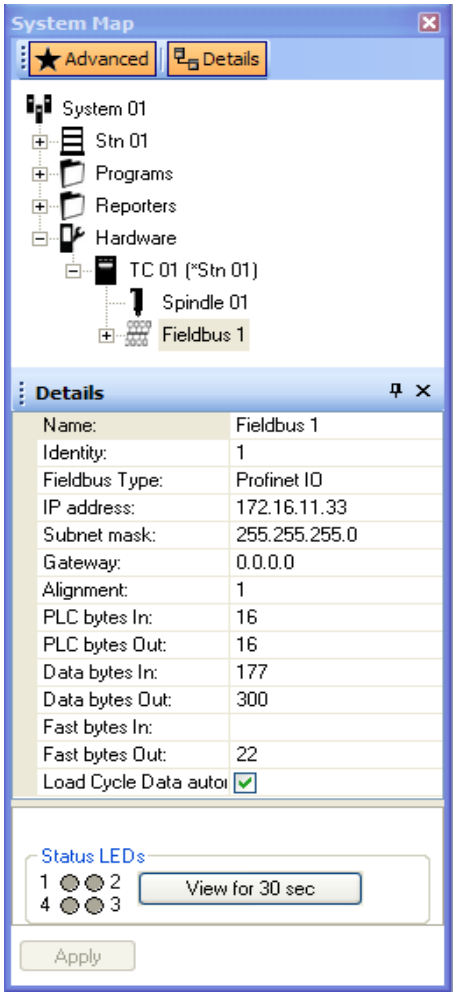
The function of the respective LED is as follows:

LED	Function	Indicates
1	**Operation Mode**	**Steady off** – Not online / No Power.
		Steady Green – On-line, data exchange.
		Flashing Green – On-line, clear.
		Flashing Red (1 flash)- Error in "Set Prm service".
		Flashing Red (2 flashes)- Error in "Chk_Cfg service".
2	**Status**	**Steady off** – No power or not initialized.
		Steady Green – Initialized.
		Flasing Green – Initialized, diagnostic event(s) present.
		Steady Red - Exception Error.

8.12.3.3 ProfiNet IO

The Anybus-CC PROFINET IO communication module provides instant PROFINET Real Time connectivity via the host interface and supports the following features:

- ProfiNet IO communication, up to 100Mbps full duplex
- Up to 256 bytes of Real Time I/O
- Up to 8192 bytes of acyclic data such as cycle data, trace etc.
- Generic GSD-file.



Use **Name** to specify the name of the device. This name should be referred to when you create a reporter for the fieldbus using the **New reporter** form.

Fieldbus Type should be set to ProfiNet IO.

Specify with **IP address** the address the AnyBus-CC module initially should have on the network. Also specify the **Subnet mask** and the IP address of an optional **Gateway**. These data can be changed by the ProfiNet IO Master when connected.

Alignment defines the size (in bytes) of the individual input and output modules that are used when the I/O area is configured on the ProfiNet master. The size of the used modules must be **equal** to the value of Alignment. Since max 64 modules can be used **Alignment** also controls the maximum size of data that can be transferred as cyclic IO data, see table below.

See chapter: **Fieldbus Interface** for a description on **PLC bytes In**, **PLC bytes Out**, **Data bytes In**, and **Data bytes Out.**

The parameter **Fast bytes In** defines the size (in bytes) of the input data that is transferred as cyclic, or real time, IO data on the ProfiNet IO network. Max value is **PLC bytes In** + **Data bytes In.** Leaving the parameter "blank" will make all input data transfererred as cyclic IO data.

Fast bytes Out defines the size of the output data that is transferred as cyclic IO data on the ProfiNet IO network. Max value is **PLC bytes Out** + **Data bytes Out**. Leaving the parameter “blank" will make all output data transfererred as cyclic IO data.

The "**fast**" areas always start at offset zero (0) of the respective input or output area. The remaining part of the areas are transferred as acyclic data on the ProfiNet IO network.

The input as well as the output area of the Profinet IO module is divided in data blocks. The maximum size of each block is 64 bytes. For example, if 600 bytes output is configured it will be available as ten blocks, the first nine contains 64 * 9 = 576 bytes and the tenth 24 bytes.

The AnyBus-CC IN and OUT areas can be configured to a maximum size according to the following table:

Alignment	Max fast bytes In/Out (Total)	Max Acyclic bytes In/Out (Total)	Note
1	64/64 (64)	8192/8192 (8192)	Mapped as an 8 bit byte array.
2	128/128 (128)	8192/8192 (8192)	Mapped as a 16 bit word array.
4	256/256 (256)	8192/8192 (8192)	Mapped as a 32 bit double word array.
8	256/256 (512)	8192/8192 (8192)	Mapped as a 64 bit quad word array.

Note: When the cyclic IO data is configured on the ProfiNet IO master side it must be done in the following order:

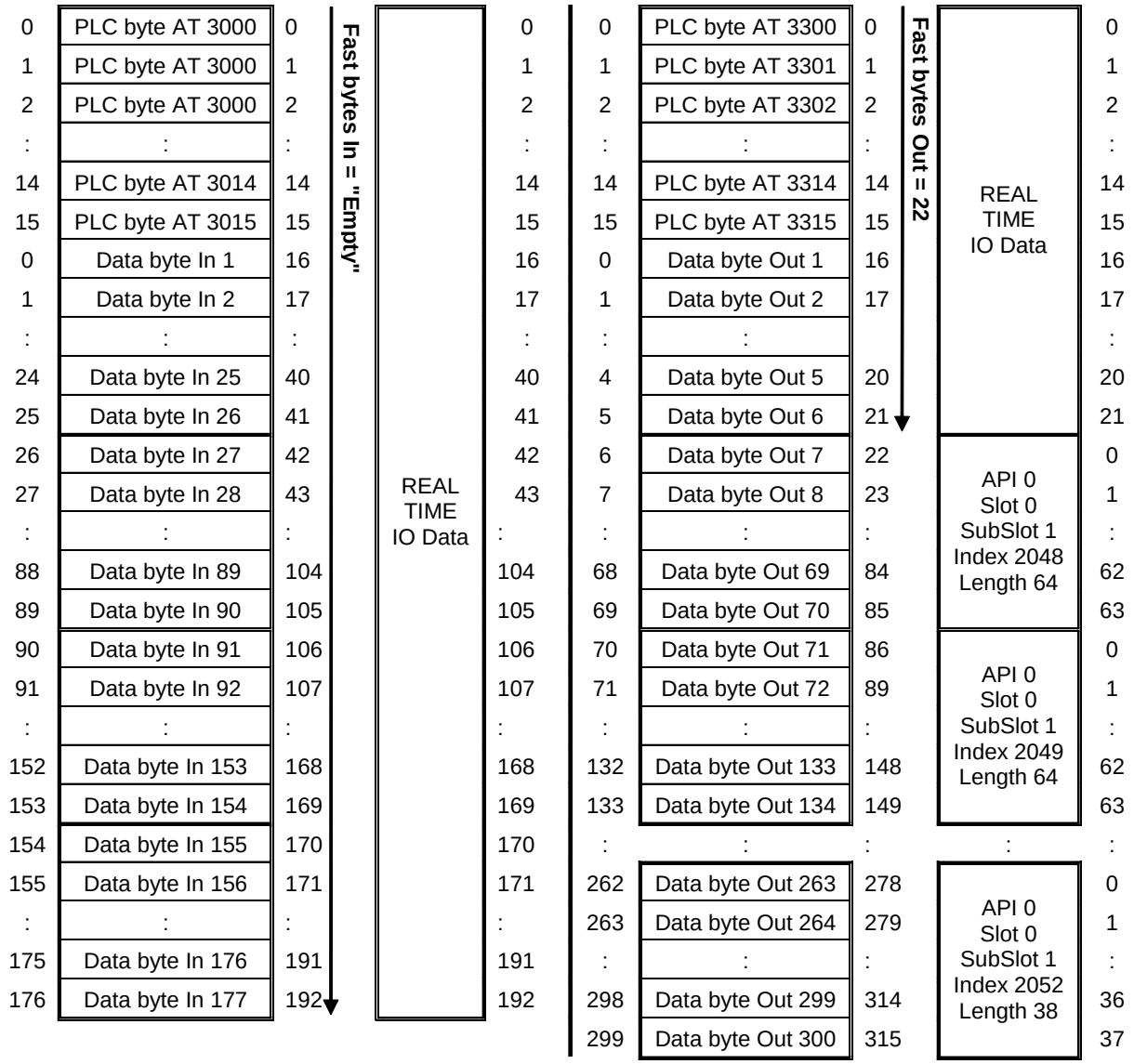
1. All data sent from the master to PowerMACS
2. All data sent from PowerMACS to the master

Acyclic data Inputs to PowerMACS are sent with “Write record” commands and acyclic data Outputs from PowerMACS are sent with “Read record” commands and are mapped to parameter data as follows:

API	Slot	Subslot	Index		Parameter block address
			Input	Output	
0	0	1	1024	2048	0 – 63
0	0	1	1025	2049	63 – 127
0	0	1	1026	2050	128 – 191
0	0	1	1027	2051	192 – 255
0	0	1	1028	2052	256 – 319
0	0	1	1029	2053	320 – 383
0	0	1	1030	2054	384 – 447
:	:	:	:		:
0	0	1	1145	2169	7744 – 7807
0	0	1	1146	2170	7808 – 7871
0	0	1	1147	2171	7872 – 7935
0	0	1	1148	2172	7936 – 7999
0	0	1	1149	2173	8000 – 8063
0	0	1	1150	2174	8064 – 8127
0	0	1	1151	2175	8128 – 8191

Note: Each index represents a parameter block that is 64 byte long but depending of the configured size of the acyclic data the last block could be less than 64 bytes.

With parameters set as in the above example the input and output areas are mapped as below:



Three LED's on the AnyBus module indicate the status of the ProfiNet IO. These LED:s can also be displayed from the **System Map** form, see **Indication of the fieldbus node's** for how to do this.

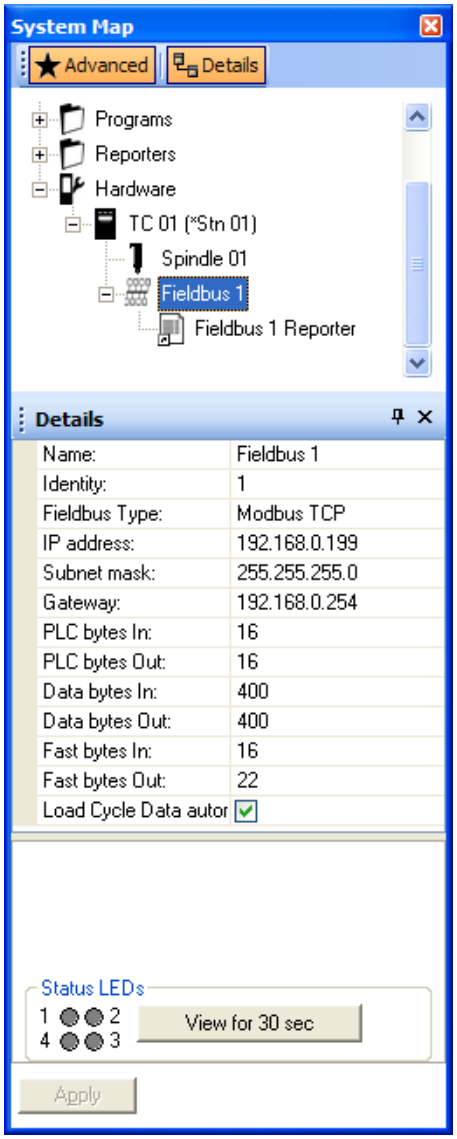
The function of the respective LED is as follows:

LED	Function	Indicates
1	**Network status**	**Steady off** – No power applied to module or no connection established
		Steady Green – Connection with IO Controller established. – IO Controller in RUN State.
		Flashing Green – Connection with IO Controller established. – IO Controller in STOP State.
2	**Module status**	**Steady off** – No power applied to module or module I SETUP state
		Steady Green – Normal operation.
		Green, 1 flash – Diagnostic event present.
		Green, 2 flashes – Used by engineering tools to identify the node on the network.
		Steady Red – Major internal error detected
		Red, 1 flash – Expected Identification differs from Read Identification.
		Red, 2 flashes – IP address not set.
		Red, 3 flashes – Station name not set.
		Red, 4 flashes – Module has encountered a major internal error.
4	**Link/Activity**	**Steady off** – No link, no communication present.
		Steady Green – Ethernet link established no communication present.
		Flashing Green – Ethernet link established communication present.

8.12.3.4 Modbus TCP

The Anybus-CC Modbus TCP communication module provides instant Modbus TCP communication via the host interface and supports the following features:

- Modbus TCP communication, with 10/100Mbit, full/half duplex operation
- Up to 256 bytes of Real Time I/O
- Up to 8192 bytes of acyclic data such as cycle data, trace etc.
- Acyclic data can be accessed from the network.



Use **Name** to specify the name of the device. This name should be referred to when you create a reporter for the fieldbus using the **New reporter** form.

Fieldbus Type should be set to Modbus TCP.

Specify with **IP address** the address the Modbus TCP module should have on the network. Also specify the **Subnet mask** and the IP address of an optional **Gateway**.

See chapter: **Fieldbus Interface** for a description on **PLC bytes In**, **PLC bytes Out**, **Data bytes In**, and **Data bytes Out.**

The parameter **Fast bytes In** defines the size (in bytes) of the input data that is transferred as cyclic, or real time, IO data on the ProfiNet IO network. Max value is **PLC bytes In** + **Data bytes In.** Leaving the parameter "blank" will make all input data transfererred as cyclic IO data.

Fast bytes Out defines the size (in bytes) of the output data that is transferred as cyclic IO data on the Modbus TCP network. Max value is **PLC bytes Out** + **Data bytes Out**. Leaving the parameter “blank" will make all output data transferred as I/O data.

The "**fast**" areas always start at offset zero (0) of the respective input or output area. The remaining part of the respective areas is transferred as acyclic data.

The input as well as the output area of the Modbus TCP module is divided in data blocks. The maximum size of each block is 32 bytes. For example, if 400 bytes output is configured it will be available as thirteen blocks, the first twelve contains 32 * 12 = 384 bytes and the thirteenth 16 bytes.

The AnyBus-S Modbus TCP communication module acts as a server and supports the following Modbus TCP commands:

Function code	Sub code	Function Name	Register type	Affects Area See note 1	Addressing method
1	-	Read coils	0	IN/OUT	Bit
2	-	Read Input discrete	1	IN/OUT	Bit
3	-	Read multiple registers	4	IN/OUT	Word
4	-	Read input registers	3	IN/OUT	Word
5	-	Write coil	0	OUT	Bit
6	-	Write single register	4	OUT	Word
15	-	Write multiple coils	0	OUT	Bit
16	-	Write multiple registers	4	OUT	Word
23	-	Read/Write multiple registers	4	IN/OUT	Word
43	14	Read Device Identification	-	-	-

Note 1: IN/OUT here refer to the client side. That is, IN is input to the client (output from PowerMACS) and OUT is output from the client (input to the PowerMACS).

The Modbus TCP IN and OUT areas can be configured to a maximum size of 8192 bytes. The input and output areas of the PowerMACS fieldbus device are available from the Modbus TCP network side as registers. Each register represents a word of data on the fieldbus device.

The Fast bytes IN area (output from PowerMACS) using 04: Input Register:

Word addr	Bit address							
0x000	0x0000	0x0001	0x0002	0x0003	0x0004	…	0x000E	0x000F
0x001	0x0010	0x0011	0x0012	0x0013	0x0014	…	0x001E	0x001F
...	...	...	...	...	...	…	...	...
0x1FFF	0x1FFF0	0x1FFF1	0x1FFF2	0x1FFF3	0x1FFF4	…	0x1FFFE	0x1FFFF

The Fast bytes OUT area (input to PowerMACS) using 03: Holding Register:

Word addr	Bit address							
0x000	0x0000	0x0001	0x0002	0x0003	0x0004	…	0x000E	0x000F
0x001	0x0010	0x0011	0x0012	0x0013	0x0014	…	0x001E	0x001F
...	...	...	...	...	...	…	...	...
0x1FFF	0x1FFF0	0x1FFF1	0x1FFF2	0x1FFF3	0x1FFF4	…	0x1FFFE	0x1FFFF

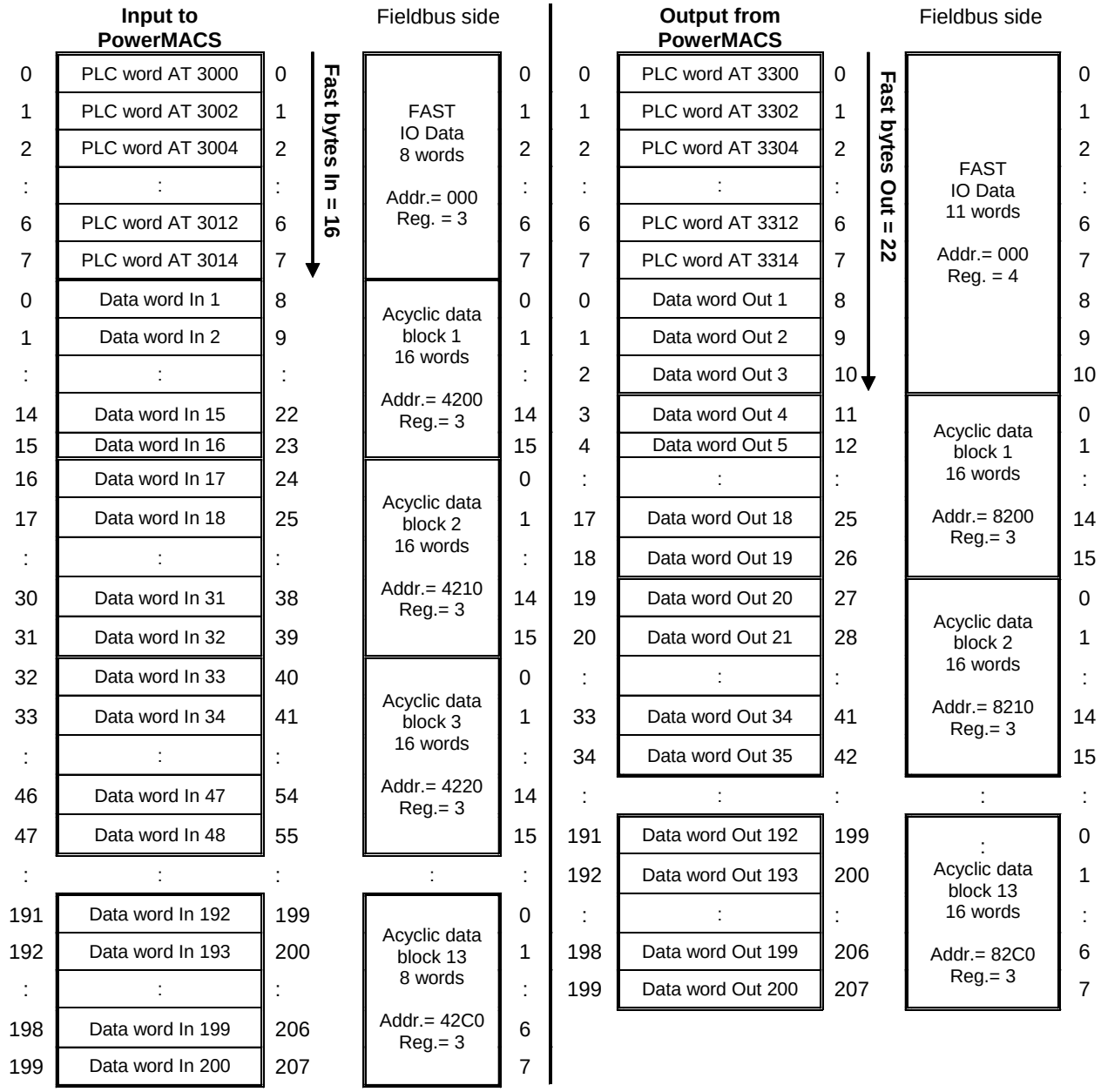
The Acyclic IN area (output from PowerMACS) using 03: Holding Register:

Word addr	Bit address							
0x8200	0x82000	0x82001	0x82002	0x82003	0x82004	…	0x8200E	0x8200F
0x8201	0x82010	0x82011	0x82012	0x82013	0x82014	…	0x8201E	0x8201F
...	...	...	...	...	...	…	...	...
0xA1FF	0xA1FF0	0xA1FF1	0xA1FF2	0xA1FF3	0xA1FF4	…	0xA1FFE	0xA1FFF

The Acyclic OUT area (input to PowerMACS) using 03: Holding Register:

Word addr	Bit address							
0x4200	0x42000	0x42001	0x42002	0x42003	0x42004	…	0x4200E	0x4200F
0x4201	0x42010	0x42011	0x42012	0x42013	0x42014	…	0x4201E	0x4201F
...	...	...	...	...	...	…	...	...
0x61FF	0x61FF0	0x61FF1	0x61FF2	0x61FF3	0x61FF4	…	0x61FFE	0x61FFF

With parameters set as in the example above the input and output areas will be mapped as below (note that he offsets are in words):



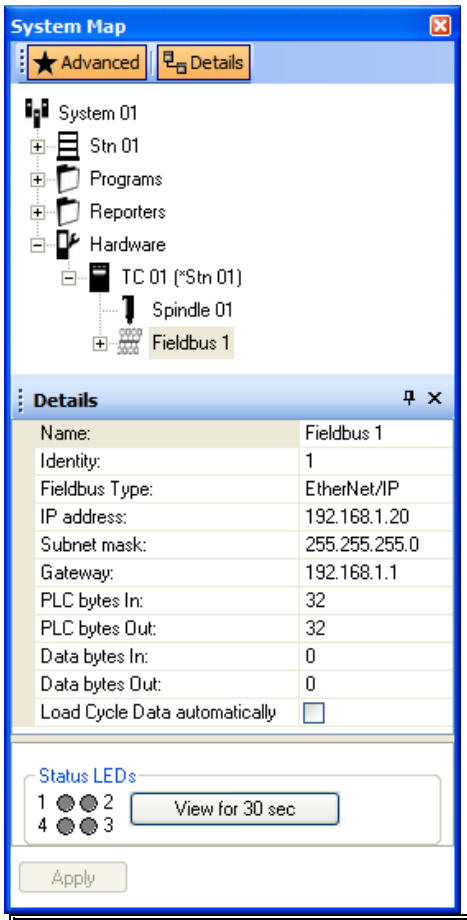
Three LED's on the AnyBus module indicate the status of the Modbus TCP. These LED:s can also be displayed from the **System Map** form, see **Indication of the fieldbus node's** for how to do this.

The function of the respective LED is as follows:

LED	Function	Indicates
1	**Network Status**	**Off** - No power or no IP address
		Green - Module is in Process Active or Idle state
		Flashing Green - Waiting for connections
		Red - Duplicate IP address, or FATAl event
		Flashing Red - Process Active Timeout
2	**Module Status**	**Off** – No power
		Green - Normal operation
		Red - Major fault; module is in state EXCEPTION(or FATAL event)
		Flashing Red - Minor fault
3	**Link/Activity**	**Off** – No link, no activity
		Green - Link established
		Flickering Green - Activity

8.12.3.5 Ethernet/IP

The System Map looks as below for a Ethernet/IP fieldbus device.



The AnyBus-CC Ethernet/IP module acts as a server and supports the Assembly Object (class id = 0x04) for I/O data input and output.

Use **Name** to specify the name of the device. This name should be referred to when you create a reporter for the fieldbus using the **New Reporter** form.

Fieldbus Type should be set to EtherNet/IP.

Specify with **IP address** the address the AnyBus-CC board should have on the network, seen from a client. Also specify the **Subnet mask** and the IP address of an optional **Gateway**.

See chapter: **Fieldbus Interface** for a description on **PLC bytes In**, **PLC bytes Out**, **Data bytes In**, and **Data bytes Out.**

The AnyBus-CC IN and OUT areas can be configured to a maximum size of 256 bytes each.

Inputs to PowerMACS (that is, the AnyBus OUT area) is mapped as follows:

Class Id	I/O or Param block No	Instance Id	Attribute Id
0x04 – Assembly Object	I/O block No 1	0x96	0x03
	I/O block No 2	0x97	
	I/O block No 3	0x98	
	I/O block No 4	0x99	

Outputs from PowerMACS (that is, the AnyBus IN area) is mapped as follows:

Class Id	I/O or Param block No	Instance Id	Attribute Id
0x04 – Assembly Object	I/O block No 1	0x64	0x03
	I/O block No 2	0x65	
	I/O block No 3	0x66	
	I/O block No 4	0x67	

The AnyBus-CC module (from version 2.05) has instances for Input only and Listen only. There are two versions for both: normal and extended. The numbering of these instances can be changed by marking **Use extended instances.**

Instance	Default	Use extended instances
Input Only	3	6
Listen Only	4	7
Input Only Extended	6	3
Listen Only Extended	7	4

This way it is possible to switch between normal and extended mode without having to change instance numbering in other equipment.

The version of the AnyBus board is presented in the System Map. If the extended numbering is wanted, and the version is 2.03 or older, please contact Atlas Copco to get a description on how to upgrade the AnyBus board firmware to a newer version.

Three LED's on the AnyBus-CC board indicate the status of the EtherNet/IP and the AnyBus card. These LED:s can also be displayed from the **System Map** form, see **Indication of the fieldbus node's status** in the manual for how to do this.

The function of the respective LED is as follows:

LED	Function	Indicates
1	**Network Status**	**Steady off** – No power or no IP address
		Flashing Green – On-line, no connections established
		Steady Green – On-line, one or more connections established (CIP Class 1 or 3)
		Flashing Red – One or more connections timed out (CIP Class 1 or 3)
		Steady Red – Duplicate IP address or FATAL error
2	**Module Status**	**Steady off** – No power
		Flashing Green – Not configured or Scanner in Idle state
		Steady Green – Controlled by a scanner in Run state
		Flashing Red – Recoverable fault(s)

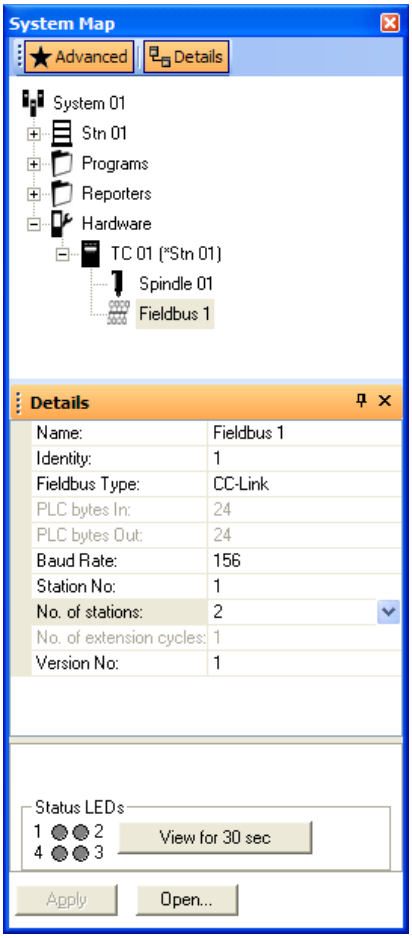
		Steady Red – Major fault (EXCEPTION-state, FATAL error)
3	**Link/Activity**	**Off** – No link, no activity
		Steady Green – Link established
		Flickering Green – Activity

8.12.3.6 CC-Link

The Anybus-CC CC-Link IO communication module provides instant CC-Link Real Time connectivity via the host interface and supports the following features:

- All profiles for a Remove Device (default PLC profile)
- Up to 256 bytes of Real Time I/O
- Selectable baud rates from 156 kbit/S to 10 Mbit/S
- Number of occupied stations: 1-4
- Support for CC.Link version 2.0 with extended data

The interface for CC-Link provides data areas for PLC Data only. It is not possible to read or write Process Data directly. Reporting result data via CC-Link can be accomplished by adding a PLC device that makes the data available in the PLC. This data can then be forwarded to the CC-Link master as normal PLC Data.



Use **Name** to specify the name of the device.

Fieldbus Type should be set to CC-Link.

The **PLC bytes In** and **PLC bytes Out** are read only and calculated automatically depending on the specified configuration. See chapter**: Fieldbus Interface** for a description on **PLC bytes In** and **PLC bytes Out**.

Use the **Baud Rate** to set the baud rate to 156 kbit/S, 625 kbit/S, 2.5 Mbit/S, 5 Mbit/S or 10 Mbit/S

Station No defines the number that identifies this station on the CC-Link network. The highest number depends on the number of stations that this node occupies.

No. of stations controls how many CC-Link stations that this node occupies on the CC-Link net. The sum of **Station No** and **No. of stations** can not exceed 65.

No. of extension cycles is available for CC-Link version 2.0 and enables the use of extended data.

VersionNo is used to set the desired CC-Link version.

The following table shows the available sizes for **PLC bytes In** and **PLC bytes Out.**

CC-Link version 1:

No. of stations	Points	Total, in bytes	PLC bytes in/Out
1	32 bits / 4 words	12	12/12
2	64 bits / 8 words	24	24/24
3	96 bits / 12 words	36	36/36
4	128 bits / 16 words	48	48/48

CC-Link version 2.0 :

	1 Extension Cycle		2 Extension Cycles		4 Extension Cycles		8 Extension Cycles	
No. of stations	Points	PLC bytes in/Out	Points	PLC bytes in/Out	Points	PLC bytes in/Out	Points	PLC bytes in/Out
1	32 bits / 4 words	12/12	32 bits / 8 words	20/20	64 bits / 16 words	40/40	128 bits / 32 words	80/80
2	64 bits / 8 words	24/24	96 bits / 16 words	44/44	192 bits / 32 words	88/88	384 bits / 64 words	176/176
3	96 bits / 12 words	36/36	160 bits / 24 words	68/68	320 bits / 48 words	136/136	640 bits / 96 words	256/256 [1]
4	128 bits / 16 words	48/48	224 bits / 32 words	92/92	448 bits / 64 words	184/184	896 bits / 128 words	256/256 [1]

[1] *When using this configuration the **PLC bytes In** and **PLC bytes Out** are limited to 256 each.*

The PLC bytes In and Out data areas is mapped so that all data bits (bit points) are placed first (beginning at offset zero) then followed by the data words (word points). See chapter: Fieldbus Interface for more information on how the input and outputs are accessed by the PLC.

Since this fieldbus type does not handle Process Data directly it is not possible to connect a Reporter to it.

With parameters set as in the above example the input and output areas is mapped as below:

byte	PowerMACS Input area	CC-Link master Output	
0	PLC bool AT 3000.0	Y	Y0 (bit 0)
:	PLC bool AT 3000.1	Data bits	Y1 (bit 1)
:	:		:
0	PLC bool AT 3000.7		Y7 (bit 7)
1	PLC bool AT 3001.0		Y8 (bit 8)
:	:		:
7	PLC bool AT 3007.7		Y63 (bit 63)
8+9	PLC word AT 3008	RWw	Word 1
10+11	PLC word AT 3010	Data word registers	Word 2
:	:		:
22+23	PLC word AT 3022		Word 8

byte	PowerMACS Output area	CC-Link master Input	
0	PLC bool AT 3300.0	X	X0 (bit 0)
:	PLC bool AT 3300.1	Data bits	X1 (bit 1)
:	:		:
0	PLC bool AT 3300.7		X7 (bit 7)
1	PLC bool AT 3301.0		X8 (bit 8)
:	:		:
7	PLC bool AT 3307.7		X63 (bit 63)
8+9	PLC word AT 3308	RWr	Word 1
10+11	PLC word AT 3310	Data word registers	Word 2
:	:		:
22+23	PLC word AT 3322		Word 8

Two LED's on the AnyBus module indicate the status of the CC-Link. These LED:s can also be displayed from the **System Map** form, see **Indication of the fieldbus node's** for how to do this.

The function of the respective LED is as follows:

LED	Function	Indicates
1	**Run**	**Steady off** – No network participation, timeout status (no power).
		Steady Green – Participating, normal operation.
		Steady Red – Major fault (FATAL error).
2	**Error**	**Steady off** – No error detected (no power).
		Steady Red – Major fault (Exception or FATAL event).
		Red, flickering – CRC error (temporary flickering).
		Red, flashing – Station Number or Baud rate has changed since startup (flashing)

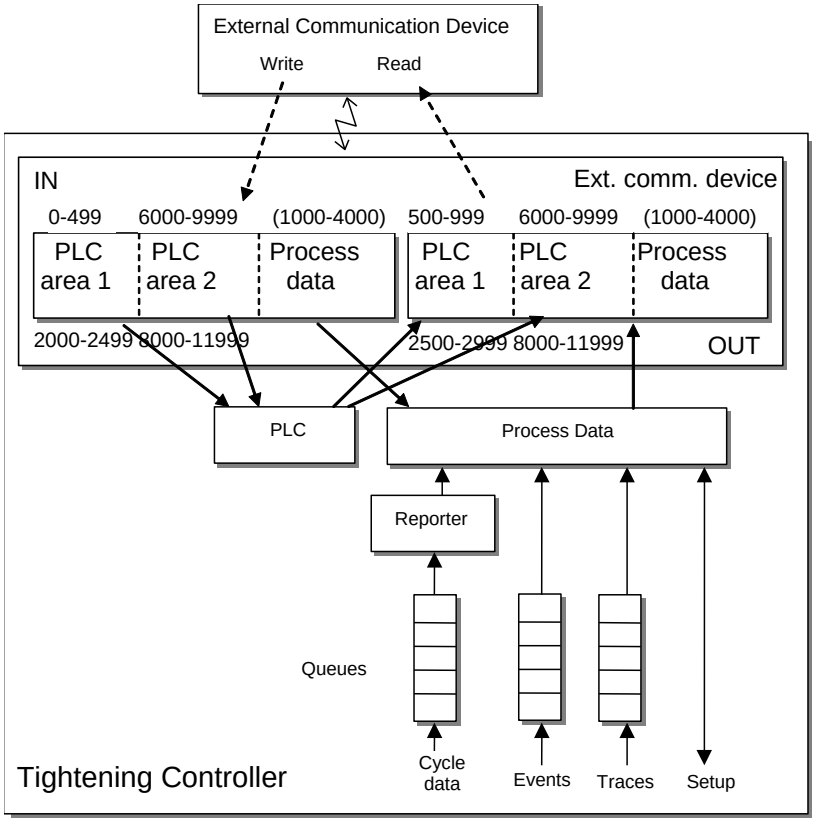
8.13 Serial Communication

The serial communication devices supports the following protocols:

- Siemens 3964R
- Telemechanique UNI-TE
- SATT Comli
- Allen-Bradley Data Highway (Plus)
- JBUS

Via an external communication device you can access the following PowerMACS functions:

- Read and write PLC signals
- Read cycle data, events and traces
- Read and write complete setups



To enable access to PowerMACS via a serial protocol add an external communication device as described in previous chapter. Add it to a TC that has a PLC, i.e. the first TC in a station. Set up which **Type** of protocol to use, which port to use and port characteristics.

If you want to transfer Process data over the external communication device then you must first create a – Reporter for it. How this is done is described in the chapter: New reporter and Edit reporter.

8.13.1 Accesses from External communication device

To access data in PowerMACS external devices should perform normal read and write operations. Depending on address different data is available:

Address	Data	Length	Read/Write
0..999	Variables in PLC area 1	Length of area as defined by drivers EXTCOM_in and EXTCOM_out on the IO_Configuration worksheet	R/W
1000	Cycle data	2..Max	R/W
2000	Events	2..Max	R/W
3000	Traces	2..Max	R/W
4000	Setup	2..Max	R/W
6000-9999	Variables in PLC area 2	Length of area as defined by drivers EXTCO2_in and EXTCO2_out on the IO_Configuration worksheet	R/W

Max depends on which serial protocol that is used.

A read or write to an address between 0 and 999 will access a memory that the PLC on that TC can access, thereby providing a way to communicate with the PLC. See the chapter: Access to PLC data for how to enabled this.

The PLC area 2 at address 6000-9999 can be used for transfer of large data, e.g. formatted cycle data. The PLC must set a positive flank (0->1) in the system global UPDATE_EXTCO2 to get this area updated.

By accessing address 1000-4000 the external device can get access to Process data.

It might seem odd to write to the Cycle Data, Event and Trace queues but by writing a two-byte integer to one of the corresponding address you can control the queues as follows:

Value	Action
-2	Skips reading of the current data item in the data queue. Next read will return next data item (or the first part of it) in the queue.
-1	Flush the data queue. All items in the specified process data queue are deleted. However, the data item currently being read will not be deleted. It can safely be read also after that this command has been issued. To remove the current data item you have to issue a skip command.
0	Rewind the data queue to the oldest value data item available in the central SRAM storage.
1..32767	Here value is interpreted as an earlier received device specific sequence number (DataNo) to which the queue should be re-winded. When the queue is read the next time the item that corresponds to DataNo = Value + 1 will be returned, if it exists in the queue. If it no longer exists in the queue then the item returned will have DataNo set to Value + 2 to indicate this. For Event and Trace the DataNo is always included in the item read. For Cycle data you must include it manual using the reporter for this .

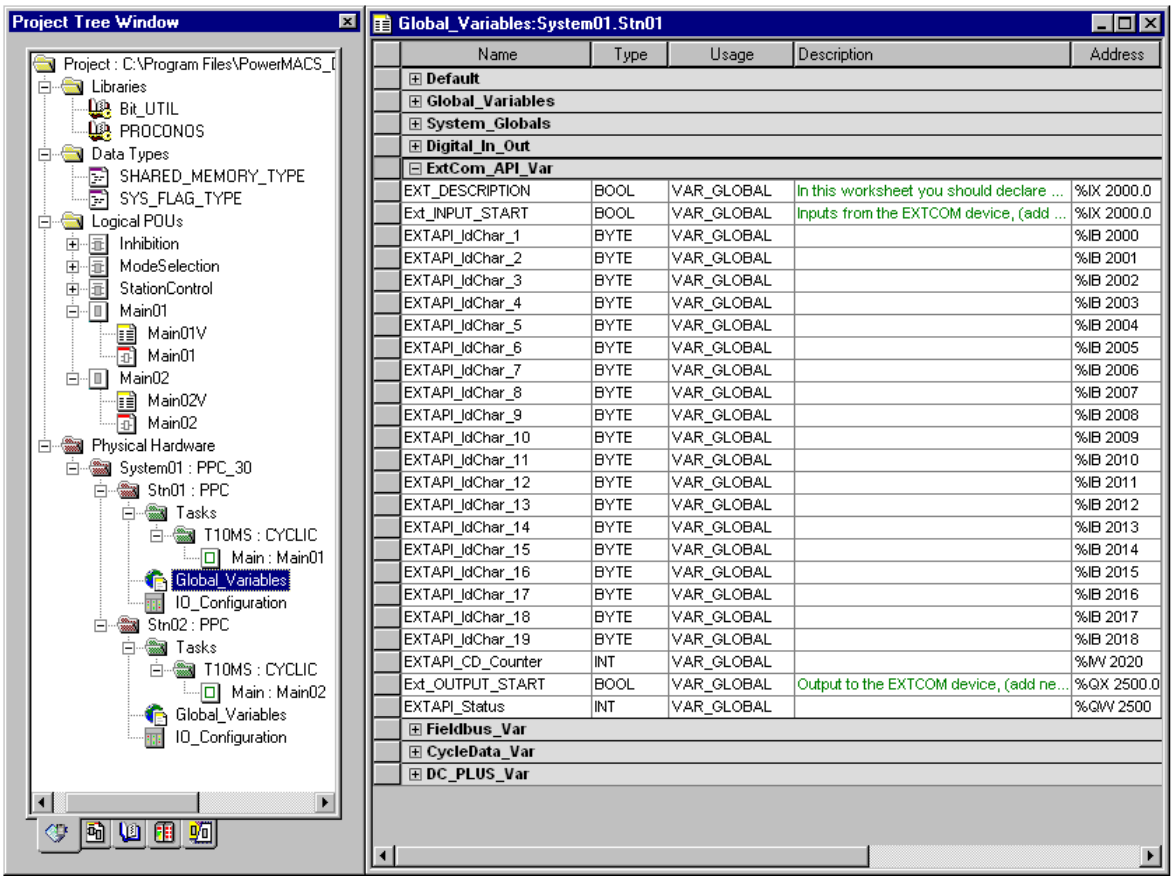
Note! If the external communication device is added to a TC that also have an API device then the PLC data area will be shared with corresponding area used by the API, Application Programmers Interface. If you use these simultaneously you must co-ordinate the use of the cells to avoid unwanted collisions.

8.13.2 Access to PLC data

The external devices accesses the PLC data by simple read and write accesses to address 0..999 and/or 8000-11999. How these respective areas are used internally is configured using the PowerMACS PLC by defining variables and connecting them with logic written using, for example, function blocks. This way the PLC programmer can set up exactly which signals to transfer via the serial protocol.

Access to the PLC part of the External communication device data area from the PowerMACS PLC is configured in two steps. First you must configure how the respective area should be used by defining variables. Secondly must configure the EXTCOM and/or EXTCO2 driver(s) to correctly map the total sizes of the respective input and output data areas defined in the first step.

The input and output areas of the fieldbus are mapped to PLC variables by declaring them in the **ExtCom_API_Var** section of the Global_Variables worksheet for the station that the device is connected to:



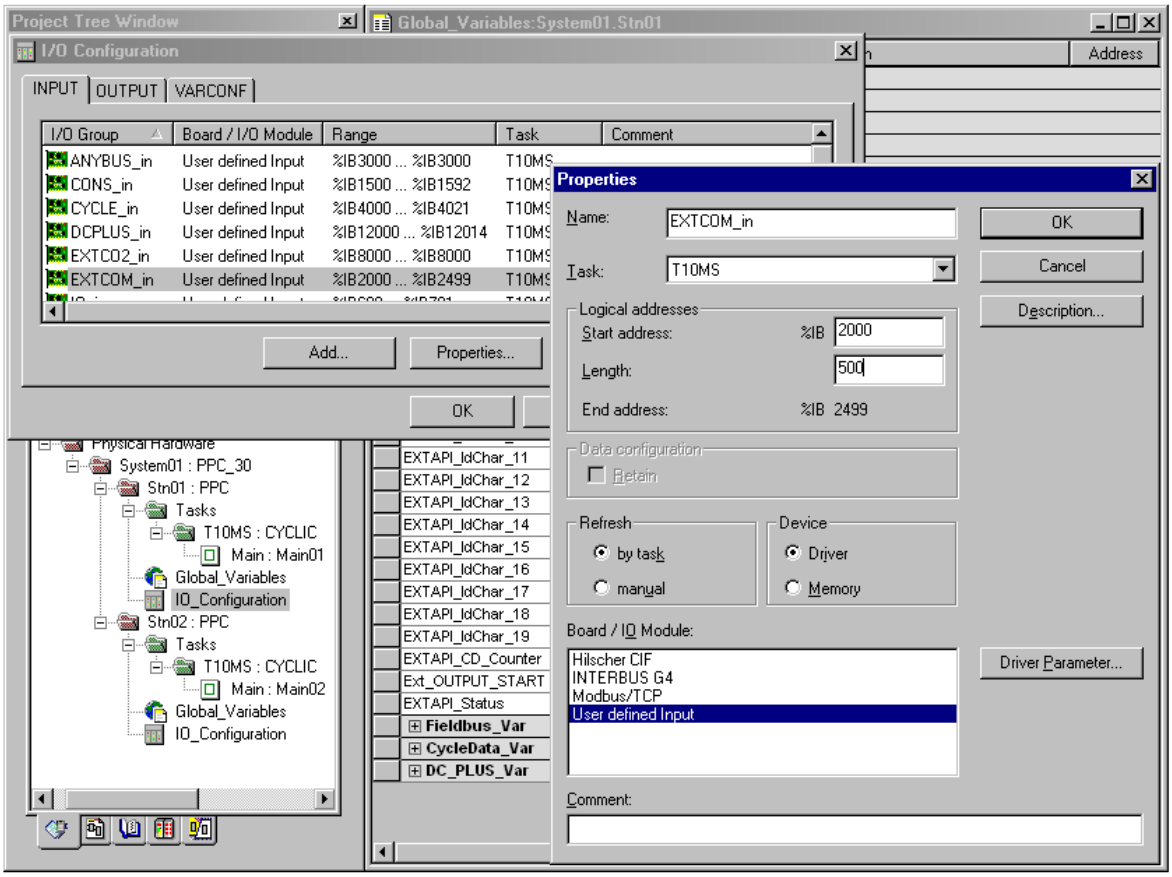
Name	Type	Usage	Description	Address
Default				
Global_Variables				
System_Globals				
Digital_In_Out				
ExtCom_API_Var				
EXT_DESCRIPTION	BOOL	VAR_GLOBAL	In this worksheet you should declare ...	%IX 2000.0
Ext_INPUT_START	BOOL	VAR_GLOBAL	Inputs from the EXTCOM device, (add ...	%IX 2000.0
EXTAPI_IdChar_1	BYTE	VAR_GLOBAL		%IB 2000
EXTAPI_IdChar_2	BYTE	VAR_GLOBAL		%IB 2001
EXTAPI_IdChar_3	BYTE	VAR_GLOBAL		%IB 2002
EXTAPI_IdChar_4	BYTE	VAR_GLOBAL		%IB 2003
EXTAPI_IdChar_5	BYTE	VAR_GLOBAL		%IB 2004
EXTAPI_IdChar_6	BYTE	VAR_GLOBAL		%IB 2005
EXTAPI_IdChar_7	BYTE	VAR_GLOBAL		%IB 2006
EXTAPI_IdChar_8	BYTE	VAR_GLOBAL		%IB 2007
EXTAPI_IdChar_9	BYTE	VAR_GLOBAL		%IB 2008
EXTAPI_IdChar_10	BYTE	VAR_GLOBAL		%IB 2009
EXTAPI_IdChar_11	BYTE	VAR_GLOBAL		%IB 2010
EXTAPI_IdChar_12	BYTE	VAR_GLOBAL		%IB 2011
EXTAPI_IdChar_13	BYTE	VAR_GLOBAL		%IB 2012
EXTAPI_IdChar_14	BYTE	VAR_GLOBAL		%IB 2013
EXTAPI_IdChar_15	BYTE	VAR_GLOBAL		%IB 2014
EXTAPI_IdChar_16	BYTE	VAR_GLOBAL		%IB 2015
EXTAPI_IdChar_17	BYTE	VAR_GLOBAL		%IB 2016
EXTAPI_IdChar_18	BYTE	VAR_GLOBAL		%IB 2017
EXTAPI_IdChar_19	BYTE	VAR_GLOBAL		%IB 2018
EXTAPI_CD_Counter	INT	VAR_GLOBAL		%IW 2020
Ext_OUTPUT_START	BOOL	VAR_GLOBAL	Output to the EXTCOM device, (add ne...	%QX 2500.0
EXTAPI_Status	INT	VAR_GLOBAL		%QW 2500
Fieldbus_Var				
CycleData_Var				
DC_PLUS_Var				

The addresses above are logical, meaning that a PLC variable declared at the logical address 2000 will correspond to address 0, that is, the first byte in the PLC input area of the external communication device.

Likewise a variable declared at the logical address 2500 is mapped to address 0 of the output area of the device. For best performance you should not define bigger areas than necessary.

When the variables are declared you are free to refer them from any POU (Program Organisation Unit) you have, or will, declare in the project.

After mapping the input and outputs variables the next step is to configure the EXTCOM and/or EXTCO2 driver(s) to copy the input and output data from the external communication device to the PLC and vice versa. This is done using the I/O Configuration dialogue which is invoked by double-clicking on the icon for **IO_Configuration** worksheet of station to which the device is connected.



In the I/O Configuration dialogue first select the INPUT tab, mark the I/O Group EXTCOM_in or EXTCO2_in and press the **Properties** button.

Change the value of **Logical addresses – Length** so that it corresponds to your needs. See **How to map input and outputs** below for a description of the possible choices.

Then configure the copying of output data by selecting the EXTCOM_out or EXTCO2_out I/O Group on the OUTPUT tab of the I/O Configuration dialogue and pressing the **Properties** button.

How to map input and outputs

The mapping between the areas inside the PowerMACS PLC and the external communication device addresses are as follows:

For the EXTCOM_in/out driver

PLC address	XCOM device address
2000..2999	0..999

For the EXTCO2_in/out driver

PLC address	XCOM device address
8000..11999	6000..9999

How much of the respective areas to use for inputs and/or outputs is set up using the drivers as explained below. Please note that if an area is used for both inputs and outputs the outputs must be located directly after the inputs.

Example:

	Correct	Erroneous
Inputs	8000..8999	8000..8999
Outputs	9000..9999	10000..10999

Note! All numerical values are written and read as Big Endian values (commonly referred to as Motorola format).

8.13.3 Access to Process data

The external device accesses the Process Data areas by normal read and writes accesses.

Since Process Data quite often is bigger than what can be transferred using a single read or write operation all process data handling are done by splitting up each process data item into smaller parts called packages. Each access will transfer a new package. The external device must then put the packages together to recreate the original data. How to do this is described in chapter: Process data.

8.13.4 Serial Communication Protocol Information

Serial communication devices use serial protocols to communicate with PowerMACS. Chapters below do not describe the protocols in full detail but gives a brief description of major characteristics and how they are used with PowerMACS.

The communication is for all protocols based on simple read and write operations. Examples of character flow is included for two types of communication:

- Read five integers at address 500..509, i.e. PLC outputs area, address 2500..2509

- Write five integers at address 0..9, i.e. PLC input area, address 2000..2009

External device is indicated with XD, PowerMACS with PM. Data that are irrelevant for PowerMACS are indicated with ‘*’.

8.13.4.1 JBUS

Two types of commands are used:

- Word read (function 3) to read data from PowerMACS
- Multiple word write (function 16) to write data in PowerMACS

Checksum is calculated as CRC-16 for all bytes from the first byte (slave no) up until the checksum.

PowerMACS will respond to all telegrams independent of Slave no from XD.

Reading 5 integers at address 500

XD	PM	Comment
0x00		* Slave no
0x03		Read command
0x01		Address, MSB
0xF4		Address, LSB
0x00		Size in words, MSB
0x05		Size in words, LSB
0xC4		Checksum, LSB
0x16		Checksum, MSB
	0x00	Slave no
	0x03	Read command
	0x0A	Number of bytes
	xx	Data
	:	5 x (MSB-LSB)
	xx	Checksum, LSB
	xx	Checksum, MSB

Writing 5 integers at address 0

XD	PM	Comment
0x00		* Slave no
0x10		Write command
0x00		Address, MSB
0x00		Address, LSB
0x00		Size in words, MSB
0x05		Size in words, LSB
0x0A		Size in bytes
xx		Data
:		5 x (MSB-LSB)
xx		Checksum, LSB
xx		Checksum, MSB
	0x00	Slave no
	0x10	Write command
	0x00	Address, MSB
	0x00	Address, LSB
	0x00	Size in words, MSB
	0x05	Size in words, LSB
	0x01	Checksum, LSB
	0xDB	Checksum, MSB

8.13.4.2 Siemens 3964R

Two types of commands are used:

- A-telegram for writing data to PowerMACS
- E-telegram for reading data from PowerMACS

Checksum is calculated as XOR for all bytes from the first byte after STX up until and including ETX.

Reading 5 integers at address 500

XD	PM	Comment
0x02		STX
	0x10	DLE
0x00		*
0x00		*
0x45		'E'
0x44		'D'
0x01		Address, MSB
0xF4		Address, LSB
0x00		Size in words, MSB
0x05		Size in words, LSB
0x00		*
0x00		*
0x10		DLE
0x03		ETX
0xE2		Checksum
	0x10	DLE
	0x02	STX
0x10		DLE
	0x00	
	0x00	
	0x00	
	0x00	Error number
	xx	Data
	:	5 x (MSB-LSB)
	0x10	DLE
	0x03	ETX
	xx	Checksum
0x10		DLE

Writing 5 integers at address 0

XD	PM	Comment
0x02		STX
	0x10	DLE
0x00		*
0x00		*
0x41		'A'
0x44		'D'
0x00		Address, MSB
0x00		Address, LSB
0x00		Size in words, MSB
0x05		Size in words, LSB
0x00		*
0x00		*
xx		Data,
:		5 x (MSB+LSB)
0x10		DLE
0x03		ETX
xx		Checksum
	0x10	DLE
	0x02	STX
0x10		DLE
	0x00	
	0x00	
	0x00	
	0x00	Error number
	0x10	DLE
	0x03	ETX
	0x13	Checksum
0x10		DLE

8.13.4.3 Telemechanique UNI-TE

Five types of commands are used:

- Read object (command 0x36) reading data from PowerMACS.
- Write object (command 0x37) for writing data to PowerMACS
- Identification (command 0x0F) for identification of PowerMACS
- Protocol Version (command 0x30) for protocol version used in PowerMACS
- Mirror (command 0xFA) for performance measurement between the master and PowerMACS

For Read and Write object the supported segments and types are:

- Segment 0x68, Type 0x07 (internal word number)
- Segment 0x68, Type 0x08 (internal double word number)

Checksum is calculated as the modulo-256 sum of all bytes from and including DLE-STX up until and including the last character before the checksum.

If PowerMACS is configured with slave number = 1..255 it will respond only if Slave no from XD is the correct one. If slave number = 0 PowerMACS will respond to any Slave no from XD.

See next page for an example of using the Read object and Write object command

Reading 5 words at address 500

XD	PM	Comment
0x10		DLE
0x02		STX
0x00		Slave no
0x0E		Length of rest of data
0x14		* Type of addressing
0x01		* Network
0x02		* Station
0x03		* Gate
0x04		* Module
0x05		* Channel
0x36		Read command
0x00		* Category
0x68		* Segment
0x07		* Type
0xF4		Address LSB
0x01		Address MSB
0x05		Length LSB
0x00		Length MSB
0xD2		Checksum
	0x06	ACK
0x10		DLE
0x05		ENQ
0x00		Slave no
	0x10	DLE
	0x02	STX
	0x00	Slave no
	0x0A	Length
	0x14	Type of address
	0x01	* Network
	0x02	* Station
	0x03	* Gate
	0x04	* Module
	0x05	* Channel
	0x66	Confirm
	0x07	* (Type)
	xx	Data,
	:	5 x (MSB+LSB)
	xx	Checksum

Writing 5 words at address 0

XD	PM	Comment
0x10		DLE
0x02		STX
0x00		Slave no
0x18		Length of rest of data
0x14		* Type of addressing
0x01		* Network
0x02		* Station
0x03		* Gate
0x04		* Module
0x05		* Channel
0x37		Write command
0x00		* Category
0x68		* Segment
0x07		* Type
0x00		Address LSB
0x00		Address MSB
0x05		Length LSB
0x00		Length MSB
xx		Data,
:		5 x (MSB+LSB)
xx		Checksum
	0x06	ACK
0x10		DLE
0x05		ENQ
0x00		Slave no
	0x10	DLE
	0x02	STX
	0x00	Slave no
	0x07	Length
	0x14	Type of address
	0x01	* Network
	0x02	* Station
	0x03	* Gate
	0x04	* Module
	0x05	* Channel
	0xFE	Confirm
	0x3A	Checksum
0x06		ACK

8.13.4.4 SATT Comli

Checksum is calculated as XOR for all bytes from the first byte after STX up until and including ETX.

Data bytes have reverse bit order, i.e. bit7 = bit0, bit6 = bit1 etc. PowerMACS will respond to all telegrams independent of Destination from XD.

Reading 5 integers at address 500

XD	PM	Comment
0x02		STX
0x30		* Destination, MSB, ‘0’
0x31		* Destination, MSB, ‘1’
0x31		Stamp, ‘1’
0x32		Read command, ‘2’
0x30		Address, digit 1, ‘0’
0x31		Address, digit 2, ‘1’
0x46		Address, digit 3, ‘F’
0x34		Address, digit 4, ‘4’
0x30		Size, digit 1, ‘0’
0x41		Size, digit 2, ‘A’
0x03		ETX
0x03		Checksum
	0x02	STX
	0x30	Destination, MSB, ‘0’
	0x30	Destination, MSB, ‘0’
	0x31	Stamp, ‘1’
	0x30	‘Write’ command
	0x30	Address, digit 1, ‘0’
	0x31	Address, digit 2, ‘1’
	0x46	Address, digit 3, ‘F’
	0x34	Address, digit 4, ‘4’
	0x30	Size, digit 1, ‘0’
	0x41	Size, digit 2, ‘A’
	xx	Data,
	:	5 x (MSB+LSB)
	0x03	ETX
	xx	Checksum

Writing 5 integers at address 0

XD	PM	Comment
0x02		STX
0x30		* Destination, MSB, ‘0’
0x31		* Destination, MSB, ‘1’
0x31		Stamp, ‘1’
0x30		Write command, ‘0’
0x30		Address, digit 1, ‘0’
0x30		Address, digit 2, ‘0’
0x30		Address, digit 3, ‘0’
0x30		Address, digit 4, ‘0’
0x30		Size, digit 1, ‘0’
0x41		Size, digit 2, ‘A’
xx		Data,
:		5 x (MSB+LSB)
0x03		ETX
xx		Checksum
	0x02	STX
	0x30	Destination, MSB, ‘0’
	0x30	Destination, MSB, ‘0’
	0x31	Stamp, ‘1’
	0x31	Write confirm, ‘1’
	0x06	ACK
	0x03	ETX
	0x06	Checksum

8.13.4.5 Allen-Bradley Data Highway (Plus)

Two types of commands are supported, unprotected read for reading data from PowerMACS and unprotected write for writing data to PowerMACS. Checksum is calculated as the 2's complement of the modulo-256 sum of all bytes from the first byte after DLE-STX up until but not including DLE-ETX. PowerMACS will respond to all telegrams independent of destination DST from XD.

Reading 5 integers at address 500

XD	PM	Comment
0x10		DLE
0x02		STX
0x02		* DST
0x01		* SRC
0x01		CMD
0x00		* STS
0x05		* TNS, LSB
0x00		* TNS, MSB
0xF4		Address, LSB
0x01		Address, MSB
0x0A		Size in bytes
0x10		DLE
0x03		ETX
0xF8		Checksum
	0x10	DLE
	0x06	ACK
	0x10	DLE
	0x02	STX
	0x01	DST
	0x02	SRC
	0x41	CMD
	0x00	STS
	0x05	TNS, LSB
	0x00	TNS, MSB
	xx	Data,
	:	5 x (LSB+MSB)
	0x10	DLE
	0x03	ETX
	xx	Checksum
0x10		DLE
0x06		ACK

Writing 5 integers at address 0

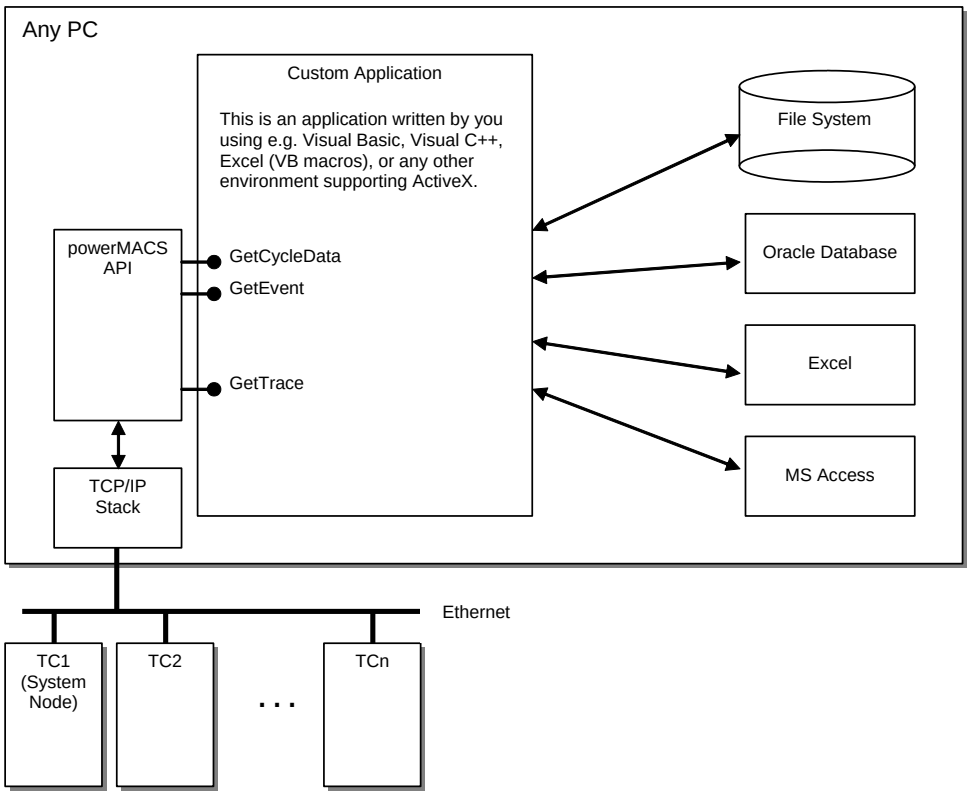
XD	PM	Comment
0x10		DLE
0x02		STX
0x02		* DST
0x01		* SRC
0x08		CMD
0x00		* STS
0x05		* TNS, LSB
0x00		* TNS, MSB
0x00		Address, LSB
0x00		Address, MSB
0x0A		Size in bytes
xx		Data,
:		5 x (LSB+MSB)
0x10		DLE
0x03		ETX
xx		Checksum
	0x10	DLE
	0x06	ACK
	0x10	DLE
	0x02	STX
	0x01	DST
	0x02	SRC
	0x48	CMD
	0x00	STS
	0x05	TNS, LSB
	0x00	TNS, MSB
	0x10	DLE
	0x03	ETX
	0xB0	Checksum
0x10		DLE
0x06		ACK

8.14 API, Application Programmers Interface

The *API, Application Programmers Interface,* is an interface to the PowerMACS TC system that makes it possible to access data in the PowerMACS TC from, externally written, custom applications.

The PowerMACS API is a software library that serves as the interface between the custom application and the PowerMACS TC. This means that it exports a number of functions that the custom application can call in order to access the PowerMACS target system without needing to know the details on how they are performed. Today the PowerMACS API exports functions for performing the following tasks:

- Read and write PLC signals from/to the PowerMACS station PLC
- Read cycle data from the PowerMACS system
- Read events from the PowerMACS system
- Read traces from the PowerMACS system
- Read and write complete setups from/to PowerMACS system
- Read or write individual parameters in a setup stored in a PowerMACS system



The PowerMACS API handles all communication needed between the PC, on which the custom application resides, and the PowerMACS TC targets involved. The developer of the custom application needs only to know which function(s) to call, and in some cases how the returned data is formatted.

When the custom application invokes one of the API functions the API first translates the request to commands understood by the target TC. These are then sent to the target TC via the TCP/IP connection. Thereafter the API waits for the answer to be returned. When the answer is received it is interpreted and in some cases translated to a more easily understood format before the API returns the answer to the custom application.

Custom applications using the API can be run on any PC that is connected to a PowerMACS TC via TCP/IP. It need not be the Console Computer, that is, the PC running ToolsTalk PowerMACS. However, only one custom application may use a specific API instance of a specific TC system at the time. The only information needed in order to establish a connection between the API and the specific target system is the IP-address and Port No (Primary/Secondary instance) of a TC in the system executing a station function.

Technically the PowerMACS API is implemented as a Microsoft ActiveX component. The main advantage with having the API implemented as an ActiveX component is that it provides a standardized way of describing its public interface. This makes it possible for a developer to use more or less any of the existing development environments that runs on a Windows machine when creating applications.

Before using the PowerMACS API you must install the component on the PC you are going to use it from. Executing the PowerMACS API install kit distributed on the same CD as the ToolsTalk PowerMACS install kit does this.

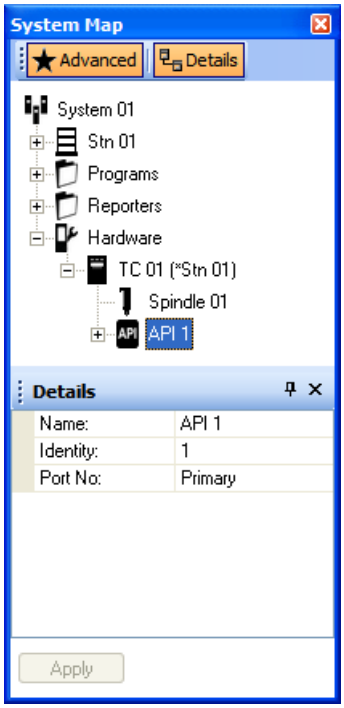
The install kit will copy the component to the computer in question and make all necessary registrations. It also installs two small custom application examples, one written in Visual Basic and one written as a VBA macro in Excel, which shows you how to use the API. Given that you have access to a running PowerMACS TC you can also use it to verify that the component was correctly installed.

After installing the PowerMACS API you can start to develop applications performing tasks not covered by the standard PowerMACS system. Typical cases are:

- Storing cycle data on disk, or in a data base, in any customer specific format
- Sending cycle data to an external SPC program
- Sending or receiving data via any serial protocol
- Backing up setups on a central server
- Etc.

To access a PowerMACS system via the API the target system in question must be set up to allow access via the API. Adding an API device to the first TC, or any other TC executing a station function, in the system enables this access. See chapter: Add a device for how to do this.

Peripheral Devices



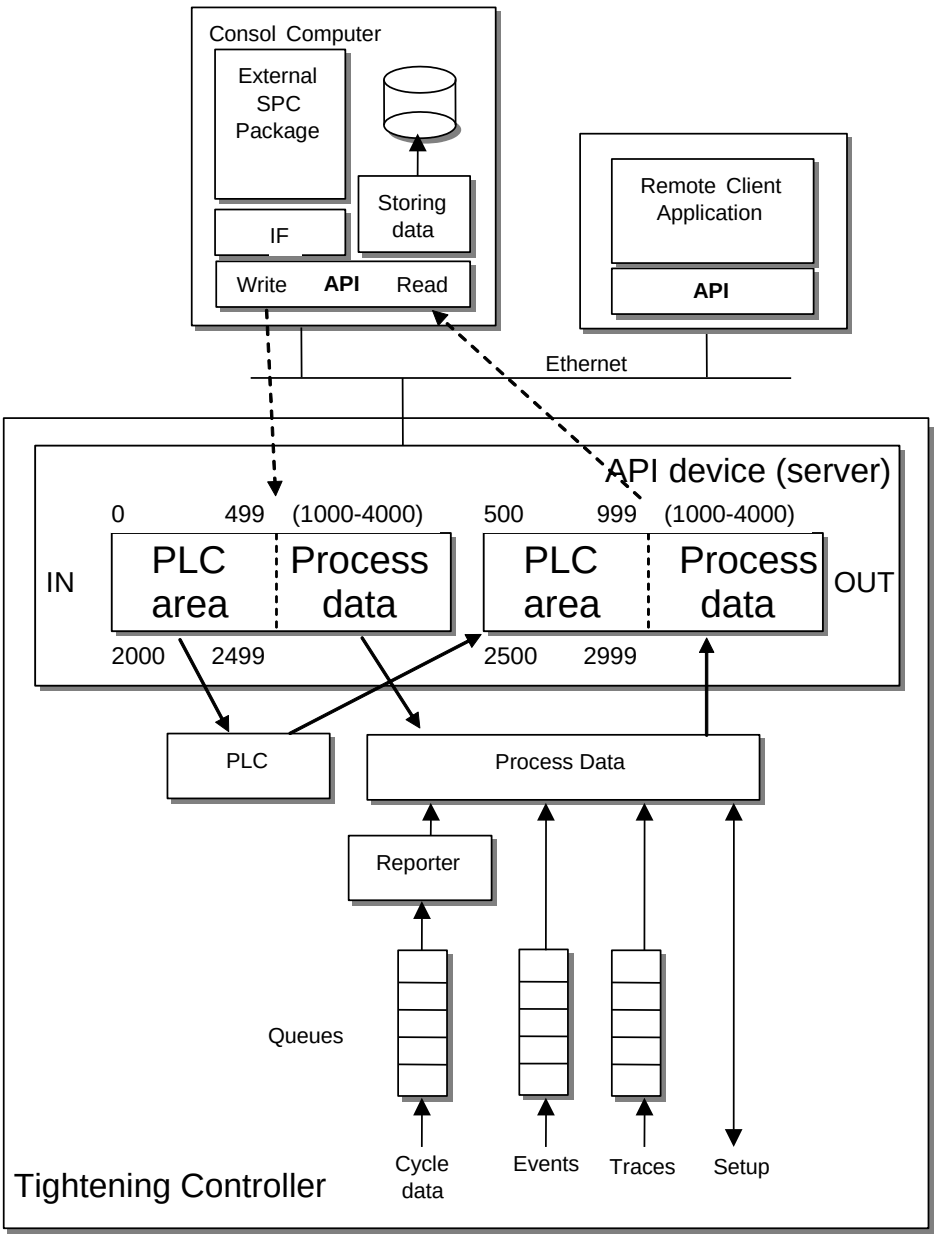
An API device can be located on the TCs running a station controller. Each of these TCs may have up to two API devices making it possible for two clients to access the same cycle data, traces, etc. If more than one device is configured on a particular TC the value of the parameter **Port No** must be used by the client software to distinguish between them. This is done using the property **DeviceInstance** of the Api object.

Before being able to access signals in one of the PowerMACS station PLCs via the API you must enable the input and output areas in the PLC. How this is done is described in the chapter: Access to PLC data.

Note! The PLC input and output data areas are shared with the areas used by the external communication devices. This means that if an external communication devices is located on TC 1 then the PLC data areas that this device accesses is the same as the area accessed via the API interface. If you use these simultaneously you must co-ordinate the use of the cells to avoid unwanted collisions.

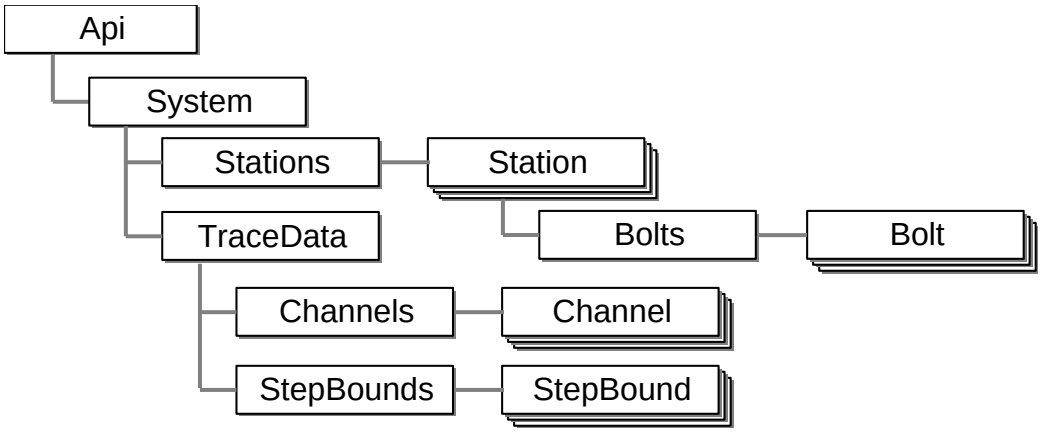
If you want to fetch Process data, that is cycle data, events, traces, etc., then you must also add a Reporter that refers to the API device. Use the Reporter to configure if you want cycle data, events and/or traces to be accessible from the API. Also, define how cycle data should be formatted, if data is sent in readable format, i.e. as characters, or in binary format. It is important to understand that it is the configuration of this reporter that controls the layout and format of the cycle data returned by the API methods GetCycleData and GetCycleDataBin.

Note! When your client application first instantiate an Api object it can take some time for the PowerMACS API ActiveX server to start up. If you are experience any problems accessing a property/method of the newly created Api instance you should make your application wait some time for the server to be started.


Consol Computer
External SPC Package
Storing data
IF
Write
API
Read
Remote Client Application
API
Ethernet
API device (server)
0
499
(1000-4000)
500
999
(1000-4000)
IN
PLC area
Process data
PLC area
Process data
OUT
2000
2499
2500
2999
PLC
Process Data
Reporter
Queues
Cycle data
Events
Traces
Setup
Tightening Controller

8.14.1 Object model

The below picture describes how the objects of the PowerMACS API are related to each other:



8.14.2 Enumerators exported by the API

The PowerMACS API exports a number of enumerated types representing return values, possible control strategies and statuses.

RetCodeEnum

This enumerator lists the codes being returned by the functions of the API

Constant	Description
eRetOk = 0	Function executed OK
eRetCmdNotAllowed = -20	Command not allowed
eRetNotOk = -51	Function did not execute OK for reasons other then listed below
eRetTimeout = -52	No answer was received from the TC within 4 seconds
eRetBufferToSmall = -53	The buffer supplied is not big enough for the returned data
eRetIpAddress = -54	IP–address is invalid
eRetConnectError = -55	A TCP/IP connection cannot be established to the specified IP-address
eRetWrongPlcAddress = -56	Incorrect PLC address (check parameters Address and/or Bit)
eRetNoData = -57	The data requested (Cycle data, Event, Trace or Setup) is not available.
eRetNotSupported = -58	Command not supported by the current version of TC SW
eRetWrongDataType = -59	Supplied data type not supported

SeverityEnum

This enumerator lists the possible Severities of an event (see View Event Log for a description of Severity).

Constant	Description
eSevInfo = 0	The event is classified as information only.
eSevWarning = 1	The event is classified as a warning.
eSevError = 2	The event is classified as an error.

DeviceInstanceEnum

This enumerator is used to select which of the API devices to access if more than one device is defined on a TC.

Constant	Description
eDevPrimary = 0	Communicate with target TC's primary API device
eDevSecondary = 1	Communicate with target TC's secondary API device

8.14.3 Api object

The **Api** object represents the root of the PowerMACS API and is the only object in the interface that can be created, or instantiated, by your client application.

Using the **Api** object your application get a description of the PowerMACS system that the **Api** currently is connected to, read and write PLC data and access Process data, such as Cycle data, Trace and Events.

For any method of the **Api** object to function you must be connected to a PowerMACS target system. Which TC to connect to is controlled using the property **IpAddress**. Normally you connect to the system node, that is, the System TC. However, in order to access the PLC of another station but the first (which always executes on the System TC) you must connect to the TC on which the station in question runs.

The following methods do only function if connected to the System TC:

If called when not connected to a System TC they will return **eRetCmdNotAllowed**.

- GetSetup
- SetSetup
- GetSetupItem
- SetSetupItem

8.14.3.1 Properties

The following table lists all properties on the **Api** object:

Property Name	Return Type	Description
IpAddress	**String**	This is the IP-address of the target that the Api will communicate to. It should be the IP-address of the System TC a Station TC. Must be set prior to any other call to an API-method.
DeviceInstance	**DeviceInstanceEnum**	This property points out which API device on the target that the API should communicate with. Should correspond to the value of the parameter "Port No" on the targets API device. Its default value is eDevPrimary.

The syntax for accessing the properties of the **Api** object is as follows:

object.**PropertyName**

where *object* is an expression that evaluates to an object of type **Api**.

8.14.3.2 GetPLCBoolEx

Description: This method reads a Boolean value from the PLC output data area for ExtCom_API.

Return type: A **RetCodeEnum** value.

Syntax: object.**GetPLCBoolEx**(Station, Address, Bit, Value)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Station	Integer	In	Specifies which stations PLC to access. Range 0 – 15 where 0 correspond to the station connected to the API device.
Address	Integer	In	Specifies the offset in the PLC area of the Byte to read. Range 0 – 999.
Bit	Integer	In	Specifies which bit to read within the byte. Range 0 – 7 where 0 is the Least significant bit and 7 is the Most significant bit.
Value	Boolean	Out	Returns the result. True if the read bit is 1, False otherwise.

8.14.3.3 SetPLCBoolEx

Description: This method writes a Boolean value to the PLC input data area for ExtCom_API.

Return type: A **RetCodeEnum** value.

Syntax: object.**SetPLCBoolEx** (Address, Bit, Value)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Station	Integer	In	Specifies which stations PLC to access. Range 0 – 15 where 0 correspond to the station connected to the API device.
Address	Integer	In	Specifies the offset in the PLC area of the Byte to write. Range 0 – 999.
Bit	Integer	In	Specifies which bit to write within the byte. Range 0 – 7 where 0 is the Least significant bit and 7 is the Most significant bit.
Value	Boolean	In	If True then the bit is set to 1. If False then it is set to 0.

8.14.3.4 GetPLCByteEx

Description: This method reads a BYTE value from the PLC output data area for ExtCom_API.

Return type: A **RetCodeEnum** value.

Syntax: object.**GetPLCByteEx**(Station, Address, Value)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Station	Integer	In	Specifies which stations PLC to access. Range 0 – 15 where 0 correspond to the station connected to the API device.
Address	Integer	In	Specifies the offset in the PLC area of the Byte to read. Range 0 – 999.
Value	Byte	Out	Returns the result. Range 0 to 255.

8.14.3.5 SetPLCByteEx

Description: This method writes a BYTE value to the PLC input data area for ExtCom_API.

Return type: A **RetCodeEnum** value.

Syntax: object.**SetPLCByteEx**(Station, Address*, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Station	Integer	In	Specifies which stations PLC to access. Range 0 – 15 where 0 correspond to the station connected to the API device.
Address	Integer	In	Specifies the offset in the PLC area of the Byte to write. Range 0 – 999.
Value	Byte	In	The data to write. Range 0 to 255.

8.14.3.6 GetPLCIntEx

Description: This method reads an Integer value from the PLC output data area for ExtCom_API.

Return type: A **RetCodeEnum** value.

Syntax: object.**GetPLCIntEx**(Station, Address*, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Station	Integer	In	Specifies which stations PLC to access. Range 0 – 15 where 0 correspond to the station connected to the API device.
Address	Integer	In	Specifies the offset in the PLC area of the first byte of the Integer to read. Range 0 – 999.
Value	Integer	Out	The read data interpreted as an Integer. Range –32768 to 32767.

8.14.3.7 SetPLCIntEx

Description: This method writes an Integer value to the PLC input data area for ExtCom_API.

Return type: A **RetCodeEnum** value.

Syntax: object.**SetPLCIntEx**(Station, Address*, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Station	Integer	In	Specifies which stations PLC to access. Range 0 – 15 where 0 correspond to the station connected to the API device.
Address	Integer	In	Specifies the offset in the PLC area of the first byte of the Integer to write. Range 0 – 999.
Value	Integer	In	The data to write. Range –32768 to 32767.

8.14.3.8 GetPLCRealEx

Description: This method reads a Real value from the PLC output data area for ExtCom_API.

Return type: A **RetCodeEnum** value.

Syntax: object.**GetPLCRealEx**(Station, *Address, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Station	Integer	In	Specifies which stations PLC to access. Range 0 – 15 where 0 correspond to the station connected to the API device.
Address	Integer	In	Specifies the offset in the PLC area of the first byte of the Real to read. Range 0 – 999.
Value	Single	Out	The read data interpreted as a Real. Range +/- 1.18 * 10^{38}.

8.14.3.9 SetPLCRealEx

Description: This method writes a Real value to the PLC input data area for ExtCom_API.

Return type: A **RetCodeEnum** value.

Syntax: object.**SetPLCRealEx**(Station, Address*, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Station	Integer	In	Specifies which stations PLC to access. Range 0 – 15 where 0 correspond to the station connected to the API device.
Address	Integer	In	Specifies the offset in the PLC area of the first byte of the Real to write. Range 0 – 999.
Value	Single	In	The data to write. Range +/- 1.18 * 10^{38}.

8.14.3.10 GetPLCStringEx

Description: This method reads 40 consecutive bytes from the PLC output data area for ExtCom_API. The read data is returned as an ASCII string.

Note! Variables declared as STRING in the PLC are prefixed by two Short integers (four bytes) where the first short specifies max possible length of the string calculated as Max length – 80. The second short defines the strings current length. That is, the first character of the string is located at offset 4 (fourth byte). All PLC strings are also terminated with a NULL (0) character. This method **does not** handle the four leading bytes.

Example: To read the string declared as below from PowerMACS station 1

```
MyString   AT %QB 2600   : STRING40;
```

you should call: `MyApi.GetPLCStringEx(1, 2604, Value)`

Return type: A **RetCodeEnum** value.

Syntax: object.**GetPLCStringEx**(Station, Address, Value)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Station	Integer	In	Specifies which stations PLC to access. Range 0 – 15 where 0 correspond to the station connected to the API device.
Address	Integer	In	Specifies the offset in the PLC area of the first byte to read. Range 0 – 999.
Value	String	Out	The read data interpreted as a string.

8.14.3.11 SetPLCStringEx

Description: This method writes the specified string to the PLC input data area for ExtCom_API as 40 consecutive bytes. Each byte will be set to the ASCII value of the respective character. See also the note for SetPLCString.

Return type: A **RetCodeEnum** value.

Syntax: object.**SetPLCStringEx**(Station, Address, Value)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Station	Integer	In	Specifies which stations PLC to access. Range 0 – 15 where 0 correspond to the station connected to the API device.
Address	Integer	In	Specifies the offset in the PLC area of the first byte to write to. Range 0 – 999.
Value	String	In	The data to write.

8.14.3.12 GetPLCBool

Description: This method is replaced with the GetPLCBoolEx method and should not be used in new applications. It is here only for backward compatibility reasons.

This method reads a Boolean value from the PLC output data area for ExtCom_API (see Access to PLC data).

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetPLCBool**(*Address, Bit, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Address	Integer	In	Specifies the offset in the PLC area of the Byte to read. Range 0 – 999.
Bit	Integer	In	Specifies which bit to read within the byte. Range 0 – 7 where 0 is the Least significant bit and 7 is the Most significant bit.
Value	Boolean	Out	Returns the result. True if the read bit is 1, False otherwise.

8.14.3.13 SetPLCBool

Description: This method is replaced with the SetPLCBoolEx method and should not be used in new applications. It is here only for backward compatibility reasons.

This method writes a Boolean value to the PLC input data area for ExtCom_API (see Access to PLC data).

Return type: A **RetCodeEnum** value.

Syntax: *object*.**SetPLCBool**(*Address, Bit, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Address	Integer	In	Specifies the offset in the PLC area of the Byte to write. Range 0 – 999.
Bit	Integer	In	Specifies which bit to write within the byte. Range 0 – 7 where 0 is the Least significant bit and 7 is the Most significant bit.
Value	Boolean	In	If True then the bit is set to 1. If False then it is set to 0.

8.14.3.14 GetPLCByte

Description: This method is replaced with the GetPLCByteEx method and should not be used in new applications. It is here only for backward compatibility reasons.

This method reads a BYTE value from the PLC output data area for ExtCom_API (see Access to PLC data).

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetPLCByte**(*Address, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Address	Integer	In	Specifies the offset in the PLC area of the Byte to read. Range 0 – 999.
Value	Byte	Out	Returns the result. Range 0 to 255.

8.14.3.15 SetPLCByte

Description: This method is replaced with the SetPLCByteEx method and should not be used in new applications. It is here only for backward compatibility reasons.

This method writes a BYTE value to the PLC input data area for ExtCom_API (see Access to PLC data).

Return type: A **RetCodeEnum** value.

Syntax: *object*.**SetPLCByte**(*Address, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Address	Integer	In	Specifies the offset in the PLC area of the Byte to write. Range 0 – 999.
Value	Byte	In	The data to write. Range 0 to 255.

8.14.3.16 GetPLCInt

Description: This method is replaced with the GetPLCIntEx method and should not be used in new applications. It is here only for backward compatibility reasons.

This method reads an Integer value from the PLC output data area for ExtCom_API (see Access to PLC data).

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetPLCInt**(*Address, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Address	Integer	In	Specifies the offset in the PLC area of the first byte of the Integer to read. Range 0 – 999.
Value	Integer	Out	The read data interpreted as an Integer. Range –32768 to 32767.

8.14.3.17 SetPLCInt

Description: This method is replaced with the SetPLCIntEx method and should not be used in new applications. It is here only for backward compatibility reasons.

This method writes an Integer value to the PLC input data area for ExtCom_API (see Access to PLC data).

Return type: A **RetCodeEnum** value.

Syntax: *object*.**SetPLCInt**(*Address, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Address	Integer	In	Specifies the offset in the PLC area of the first byte of the Integer to write. Range 0 – 999.
Value	Integer	In	The data to write. Range –32768 to 32767.

8.14.3.18 GetPLCReal

Description: This method is replaced with the GetPLCRealEx method and should not be used in new applications. It is here only for backward compatibility reasons.

This method reads a Real value from the PLC output data area for ExtCom_API (see Access to PLC data).

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetPLCReal**(*Address, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Address	Integer	In	Specifies the offset in the PLC area of the first byte of the Real to read. Range 0 – 999.
Value	Single	Out	The read data interpreted as a Real. Range +/- 1.18 * 10^{38}.

8.14.3.19 SetPLCReal

Description: This method is replaced with the SetPLCRealEx method and should not be used in new applications. It is here only for backward compatibility reasons.

This method writes a Real value to the PLC input data area for ExtCom_API (see Access to PLC data).

Return type: A **RetCodeEnum** value.

Syntax: *object*.**SetPLCReal**(*Address, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Address	Integer	In	Specifies the offset in the PLC area of the first byte of the Real to write. Range 0 – 999.
Value	Single	In	The data to write. Range +/- 1.18 * 10^{38}.

8.14.3.20 GetPLCString

Description: This method is replaced with the GetPLCStringEx method and should not be used in new applications. It is here only for backward compatibility reasons.

This method reads 40 consecutive bytes from the PLC output data area for ExtCom_API (see Access to PLC data). The read data is returned as an ASCII string.

See note in the description of GetPLCStringEx for limitations and an example on how to call.

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetPLCString**(*Address, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Address	Integer	In	Specifies the offset in the PLC area of the first byte to read. Range 0 – 999.
Value	String	Out	The read data interpreted as a string.

8.14.3.21 SetPLCString

Description: This method is replaced with the SetPLCStringEx method and should not be used in new applications. It is here only for backward compatibility reasons.

This method writes the specified string to the PLC input data area for ExtCom_API (see Access to PLC data) as 40 consecutive bytes. Each byte will be set to the ASCII value of the respective character. See also the note for SetPLCString.

Return type: A **RetCodeEnum** value.

Syntax: *object*.**SetPLCString**(*Address, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Address	Integer	In	Specifies the offset in the PLC area of the first byte to write to. Range 0 – 999.
Value	String	In	The data to write.

8.14.3.22 GetCycleData

Description: Call this method to read the next Cycle data from the FIFO queue for the API device. The data returned is interpreted as a string. Therefore this method should be used together with a Reporter configured to format data Printable.

Note 1: The prompter text, or variable name, included for a variable by checking the "Text" column in the Reporter is language dependent.

Note 2: The reporter executes on the TC and that it can only be configured using ToolsTalk PowerMACS.

The FIFO queue can hold max 500 cycle data. If overflowed then the oldest data is overwritten. Once a cycle data has been read by a call to this method it cannot be read again. If the queue is empty when called then *Buf* is set to an empty string and **eRetNoData** is returned. If *Buf* is to small to receive the read cycle data then Size is set to 0 and **eRetBufferToSmall** is returned.

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetCycleData**(*Buf, Size*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Buf	String	Out	The string to receive the read cycle data.
Size	Long	In/Out	At call: Max length of *Buf*, At return: The length of the returned string in *Buf*.

8.14.3.23 GetCycleDataBin

Description: GetCycleDataBin performs the same function as GetCycleData with the difference that the data returned is the raw binary data received from the TC. It is **not** interpreted in any way. Therefore this method should be used together with a Reporter configured to format data Binary. See chapter: GetCycleData for details.

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetCycleDataBin**(*Buf, Size*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Buf	Byte array	Out	The byte array to receive the read cycle data.
Size	Long	In/Out	At call: Max length of *Buf*, At return: The length of the returned data in *Buf*.

8.14.3.24 GetEvent

Description: This method is replaced with the GetEventEx method and should not be used in new applications. It is here only for backward compatibility reasons.

GetEvent performs the same function as GetEventEx with the difference that it **does not** return the Severity of the event.

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetEvent**(*DataNo, SeqNo, Ts, Code, Typ, Stn, Blt, Pgm, Str*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
DataNo	Integer	Out	A device unique sequence number. This number is incremented by one for each event reported over this device.
SeqNo	Long	Out	A global sequence number. This number is incremented by one for each event generated by the system.
Ts	Long	Out	Date and time for event. Expressed as no of seconds since 1970-01-01 00:00:00
Code	Integer	Out	Event code, see List of events for possible values.
Typ	Integer	Out	Event type, see View Event Log.
Stn	Integer	Out	Station on which the event occurred, 0 if not relevant
Blt	Integer	Out	Bolt for which the event occurred, 0 if not relevant
Pgm	String	Out	Tightening program connected to the event, 0 if not relevant
Str	String	Out	Event string in the language configured using ToolsTalk PowerMACS.

8.14.3.25 GetEventEx

Description: Call this method to read the next Event from the FIFO queue for the API device.

The FIFO queue can hold max 200 events. If overflowed then the oldest event is overwritten. Once an event has been read by a call to this method it cannot be read again. If the queue is empty when called then **eRetNoData** is returned.

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetEvent**(*DataNo, SeqNo, Ts, Code, Typ, Stn, Blt, Pgm, Str*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
DataNo	Integer	Out	A device unique sequence number. This number is incremented by one for each event reported over this device.
SeqNo	Long	Out	A global sequence number. This number is incremented by one for each event generated by the system.
Ts	Long	Out	Date and time for event. Expressed as no of seconds since 1970-01-01 00:00:00
Code	Integer	Out	Event code, see List of events for possible values.
Typ	Integer	Out	Event type, see View Event Log.
Stn	Integer	Out	Station on which the event occurred, 0 if not relevant
Blt	Integer	Out	Bolt for which the event occurred, 0 if not relevant
Pgm	String	Out	Tightening program connected to the event, 0 if not relevant
Str	String	Out	Event string in the language configured using ToolsTalk PowerMACS.
Sev	SeverityEnum	Out	The Severity of the event.

8.14.3.26 GetTrace

Description: Call this method to read the next Trace from the FIFO queue for the API device. See chapter: Layout of Traces for the returned data is formatted.

The FIFO queue can hold max 100 traces. If overflowed then the oldest trace is overwritten. Once a trace has been read by a call to this method it cannot be read again. If the queue is empty when called then **eRetNoData** is returned.

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetTrace**(*Buf, Size*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Buf	Byte array	Out	The byte array to receive the read trace.
Size	Long	In/Out	At call: Max length of *Buf*, At return: The length of the returned data in *Buf*.

8.14.3.27 GetTraceEx

Description: Call this method to read the next Trace from the FIFO queue for the API device. The data is returned as a TraceData object.

The FIFO queue can hold max 100 traces. If overflowed then the oldest trace is overwritten. Once a trace has been read by a call to this method it cannot be read again. If the queue is empty when called then **eRetNoData** is returned.

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetTraceEx**(*TraceData*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
TraceData	**System**	Out	A reference that can hold the returned TraceData object. Is set to Nothing (null) if error.

8.14.3.28 GetSetup

Description: This method uploads the setup from the PowerMACS system in binary form. Such a setup can be used by the SetSetup method. If there is no setup to read when called then **eRetNoData** is returned. If not connected to a System TC then **eRetCmdNotAllowed** is returned.

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetSetup**(*Buf, Size*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Buf	Byte array	Out	The byte array to receive the uploaded setup.
Size	Long	In/Out	At call: Max length of *Buf*, At return: The length of the returned data in *Buf*.

8.14.3.29 SetSetup

Description: This method downloads a setup in binary form to the PowerMACS system. The setup must have been created by an earlier call to GetSetup. The setup must have been uploaded from a system with the same version as the one it is downloaded to. If not connected to a System TC then **eRetCmdNotAllowed** is returned.

Return type: A **RetCodeEnum** value.

Syntax: *object*.**SetSetup**(*Buf, Size*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Buf	Byte array	In	The byte array containing the setup to download.
Size	Long	In	The size of the setup to download.

8.14.3.30 GetSetupItem

Description: This method returns the value of the setup item specified by the *Item* string. The *Item* string uses a specific syntax (see Layout of Setup Item Descriptions for a description) to point out which parameter to read the value of. The value is always returned as a floating-point number. Boolean parameters are converted to a Single such that zero (0) represents False and one (1) represents True. If not connected to a System TC then **eRetCmdNotAllowed** is returned.

Return type: A **RetCodeEnum** value, or if in the interval [-101..-260] syntax errors in the *Item* string. In case of syntax error then ABS(return value) - 100 is equal to the position in the string of the first error. Example: Return value -120 corresponds to a syntax error near character 20.

Syntax: *object*.**GetSetupItem**(*Item, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Item	String	In	The string specifying which parameter to read, see Layout of Setup Item Descriptions.
Value	Single	Out	The value of the read parameter.

8.14.3.31 SetSetupItem

Description: This method sets the value of the setup item specified by the *Item* string. The *Item* string uses a specific syntax (see Layout of Setup Item Descriptions for a description) to point out which parameter to write to. The value must be a floating-point number. For Boolean parameters - set zero (0) if False and one (1) if True. If not connected to a System TC then **eRetCmdNotAllowed** is returned.

Return type: A **RetCodeEnum** value, or if in the interval [-101..-260] syntax errors in the *Item* string. In case of syntax error then ABS(return value) - 100 is equal to the position in the string of the first error. Example: Return value -120 corresponds to a syntax error near character 20.

Syntax: *object*.**SetSetupItem**(*Item, Value*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
Item	String	In	The string specifying which parameter to write, see Layout of Setup Item Descriptions.
Value	Single	In	The value to write.

8.14.3.32 GetSystemDesc

Description: This method returns a System object that describes the PowerMACS system currently connected to.

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetSystemDesc**(*SysDesc*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
SysDesc	**System**	Out	A reference that can hold the returned System object. Is set to Nothing (null) if error.

8.14.3.33 GetStationDesc

Description: This method returns a Station object that describes the PowerMACS station currently connected to.

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetStationDesc**(*StnDesc*)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
StnDesc	**Station**	Out	A reference that can hold the returned Station object. Is set to Nothing (null) if error.

8.14.3.34 GetDateAndTime

Description: This method returns the current date and time of the target system.

Return type: A **RetCodeEnum** value. If an unsupported data type is used for the DateAndTime argument then **eRetWrongDataType** is returned. If the method is not supported by the connected target **eRetNotSupported** is returned. If not connected to a System TC then **eRetCmdNotAllowed** is returned.

Syntax: *object*.**GetDatyAndTime**(DateAndTime)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
DateAndTime	VARIANT	Out	The variable to receive the current date and time of the target. Since the type is VARIANT variables of different data type can be passed to the method. Currently the following are supported: • Date. (IDL data type DATE). • Long. Will return the date and time as number of seconds since 1970-01-01 00:00:00.

8.14.3.35 SetDateAndTime

Description: This method sets the current date and time of the target system.

Return type: A **RetCodeEnum** value. If an unsupported data type is used for the DateAndTime argument then **eRetWrongDataType** is returned. If the method is not supported by the connected target **eRetNotSupported** is returned. If not connected to a System TC, or if the API device is not configured as Time Server, then **eRetCmdNotAllowed** is returned.

Syntax: object.**GetDatyAndTime**(DateAndTime)

Obj./Arg.	Type	Dir	Description
object	**Api**		The Api object to operate on.
DateAndTime	VARIANT	In	The date and time to set. Since the type is VARIANT variables of different data type can be passed to the method. Currently the following are supported: • Date. (IDL data type DATE). • Long. Will return the date and time as number of seconds since 1970-01-01 00:00:00.

8.14.4 System object

The **System** object represents the PowerMACS system currently connected to.

8.14.4.1 Properties

The following table lists all properties on the **System** object:

Property Name	Return Type	Description
Name	**String**	The name of the system as entered for the System object using the System form.
Stations	**Stations**	The collection of all Station objects that exists in the PowerMACS system.

The syntax for accessing the properties of the **System** object is as follows:

object.**PropertyName**

where *object* is an expression that evaluates to an object of type **System**.

8.14.5 Stations object (collection)

The **Stations** object is a collection of Station object. It represents all station included in the PowerMACS system connect to.

8.14.5.1 Properties

The following table lists all properties on the **Stations** object:

Property Name	Return Type	Description
Count	**Long**	Returns the number of objects in the collection. Read only.

The syntax for accessing the properties of the **Stations** object is as follows:

object.**PropertyName**

8.14.5.2 Item

Description: Returns a specific Station object in the collection either by position or by the Name property of the Station object. If the value provided as *IndexKey* does not match any existing member of the collection, then Nothing (null) is returned.

Return type: A Station object.

Syntax: *object*.**Item**(*IndexKey*)

Obj./Arg.	Type	Dir	Description
object	**Stations**		The Stations object to operate on.
IndexKey	**Variant**	In	An expression that specifies the position of a member of the collection. If a numeric expression, *IndexKey* must be a number from 1 to the value of the collection's **Count** property. If a string expression, *IndexKey* must correspond to the **Name** property of a Station object.

8.14.5.3 Exists

Description: Checks if a specific Station object exists in the collection either by position or by the Name property of a Station object.

Return type: A **Boolean** value. True if the object exists, else False.

Syntax: *object*.**Exists**(*IndexKey*)

Obj./Arg.	Type	Dir	Description
object	**Stations**		The Stations object to operate on.
IndexKey	**Variant**	In	An expression that specifies the position of a member of the collection. If a numeric expression, *IndexKey* must be a number from 1 to the value of the collection's **Count** property. If a string expression, *IndexKey* must correspond to the **Name** property of a Station object.

8.14.6 Station object

The **Station** object represents a PowerMACS station.

8.14.6.1 Properties

The following table lists all properties on the **Station** object:

Property Name	Return Type	Description
Name	**String**	The name of the Station as entered for the Station object using the System form.
Number	**Integer**	The number of the station within the system [1..15].
IpAddress	**String**	The IP-address of the station (as dotted decimal).
Bolts	**Bolts**	The collection of all Bolt objects that exists within the station.

The syntax for accessing the properties of the **Station** object is as follows:

object.**PropertyName**

where *object* is an expression that evaluates to an object of type **Station**.

8.14.7 Bolts object (collection)

The **Bolts** object is a collection of Bolt objects. It represents all bolts in a particular station.

8.14.7.1 Properties

The following table lists all properties on the **Bolts** object:

Property Name	Return Type	Description
Count	**Long**	Returns the number of objects in the collection. Read only.

The syntax for accessing the properties of the **Bolts** object is as follows:

object.**PropertyName**

where *object* is an expression that evaluates to an object of type **Bolts**.

8.14.7.2 Item

Description: Returns a specific Bolt object in the collection either by position or by the Name property of the Station object. If the value provided as *IndexKey* does not match any existing member of the collection, then Nothing (null) is returned.

Return type: A Bolt object.

Syntax: *object*.**Item**(*IndexKey*)

Obj./Arg.	Type	Dir	Description
object	**Bolts**		The Stations object to operate on.
IndexKey	**Variant**	In	An expression that specifies the position of a member of the collection. If a numeric expression, *IndexKey* must be a number from 1 to the value of the collection's **Count** property. If a string expression, *IndexKey* must correspond to the **Name** property of a Bolt object.

8.14.7.3 Exists

Description: Checks if a specific Bolt object exists in the collection either by position or by the Name property of a Bolt object.

Return type: A **Boolean** value. True if the object exists, else False.

Syntax: *object*.**Exists**(*IndexKey*)

Obj./Arg.	Type	Dir	Description
object	**Bolts**		The Bolts object to operate on.
IndexKey	**Variant**	In	An expression that specifies the position of a member of the collection. If a numeric expression, *IndexKey* must be a number from 1 to the value of the collection's **Count** property. If a string expression, *IndexKey* must correspond to the **Name** property of a Bolt object.

8.14.8 Bolt object

The **Bolt** object represents a bolt within a Station.

8.14.8.1 Properties

The following table lists all properties on the **Bolt** object:

Property Name	Return Type	Description
Name	**String**	The name of the Bolt as entered using the System Map.
Number	**Integer**	The number of the bolt within the station [1..100]. First bolt is always 1.
UserBoltNo	**Integer**	The bolt number assigned to the bolt using parameter **Bolt Number** on the System Map. [1..9999].

The syntax for accessing the properties of the **Bolt** object is as follows:

object.**PropertyName**

where *object* is an expression that evaluates to an object of type **Bolt**.

8.14.9 TraceData object

The **TraceData** object contains a requested trace.

8.14.9.1 Properties

The following table lists all properties on the **TraceData** object:

Property Name	Return Type	Description
BoltId	**Long**	Bolt
Channels	**Channels**	The collection of all Channel objects that exists in the trace.
DataNo	**Integer**	A device unique sequence number. This number is incremented by one for each trace reported over the device.
DateAndTime	**Data**	Timestamp of trace in Data format
PgmName	**String**	Name of program used at the occasion
StationId	**Long**	Station
StepBounds	**StepBounds**	The collection of all StepBound objects that exists in the trace.
Ts	**Long**	Timestamp of trace

The syntax for accessing the properties of the **TraceData** object is as follows:

object.**PropertyName**

where *object* is an expression that evaluates to an object of type **TraceData**.

8.14.10 Channels object (collection)

The **Channels** object is a collection of Channel object. It represents all channels included in the requested trace.

8.14.10.1 Properties

The following table lists all properties on the **Channels** object:

Property Name	Return Type	Description
Count	**Long**	Returns the number of objects in the collection. Read only.

The syntax for accessing the properties of the **Channels** object is as follows:

object.**PropertyName**

8.14.10.2 Item

Description: Returns a specific Channel object in the collection either by position or by the Name property of the Channel object. If the value provided as *IndexKey* does not match any existing member of the collection, then Nothing (null) is returned.

Return type: A Channel object.

Syntax: *object*.**Item**(*IndexKey*)

Obj./Arg.	Type	Dir	Description
object	**Channels**		The Channels object to operate on.
IndexKey	**Variant**	In	An expression that specifies the position of a member of the collection. If a numeric expression, *IndexKey* must be a number from 1 to the value of the collection's **Count** property. If a string expression, *IndexKey* must correspond to the **Name** property of a Channel object.

8.14.10.3 Exists

Description: Checks if a specific Channel object exists in the collection either by position or by the Name property of a Channel object.

Return type: A **Boolean** value. True if the object exists, else False.

Syntax: *object*.**Exists**(*IndexKey*)

Obj./Arg.	Type	Dir	Description
object	**Channels**		The Channels object to operate on.
IndexKey	**Variant**	In	An expression that specifies the position of a member of the collection. If a numeric expression, *IndexKey* must be a number from 1 to the value of the collection's **Count** property. If a string expression, *IndexKey* must correspond to the **Name** property of a Channel object.

8.14.11 Channel object

The **Channel** object represents a channel in the trace.

8.14.11.1 Properties

The following table lists all properties on the **Channel** object:

Property Name	Return Type	Description
Name	**String**	The name of the Channel.
NoOfSamples	**Long**	The number of stored samples.
SampleTime	**Single**	The Sample time at which the stored samples were collected.
TimeOffset	**Single**	The time offset for the first stored sample, where Ts is 0.

The syntax for accessing the properties of the **Channel** object is as follows:

object.**PropertyName**

where *object* is an expression that evaluates to an object of type **Channel**.

8.14.11.2 GetSampleValueNo

Description: Give sample time as parameter and get sample number as reply.

Return type: A **Long** value. Sample number possible values are 1 to 6000.

Syntax: *object*.**GetSampleValueNo**(*SampleValueTime*)

Obj./Arg.	Type	Dir	Description
object	**Channel**		The Channel object to operate on.
SampleValueTime	**Single**	In	Time from Ts to when the current sample was taken.

8.14.11.3 GetSampleValueTime

Description: Give sample number as parameter and get sample time as reply.

Return type: A **Single** value. Time from Ts to when current sample was taken.

Syntax: *object*.**GetSampleValueTime**(*SampleValueNo*)

Obj./Arg.	Type	Dir	Description
object	**Channel**		The Channel object to operate on.
SampleValueNo	**Long**	In	Sample number that corresponds to entered sample time.

8.14.11.4 GetSampleValues

Description: Get buffer with all sample values collected for the channel.

Return type: A **RetCodeEnum** value.

Syntax: *object*.**GetSampleValues**(Buf(), Size)

Obj./Arg.	Type	Dir	Description
object	**Channel**		The Channel object to operate on.
Buf()	**Single**	Out	Array of Single, contains all sample values.
Size	**Long**	In/Out	Available array size in and stored number of samples out.

8.14.12 StepBounds object (collection)

The **StepBounds** object is a collection of StepBound object. It represents all StepBounds included in the requested trace.

8.14.12.1 Properties

The following table lists all properties on the **StepBounds** object:

Property Name	Return Type	Description
Count	**Long**	Returns the number of objects in the collection. Read only.

The syntax for accessing the properties of the **StepBounds** object is as follows:

object.**PropertyName**

8.14.12.2 Item

Description: Returns a specific StepBound object in the collection by position. If the value provided as *IndexKey* does not match any existing member of the collection, then Nothing (null) is returned.

Return type: A StepBound object.

Syntax: *object*.**Item**(*IndexKey*)

Obj./Arg.	Type	Dir	Description
object	**StepBounds**		The StepBounds object to operate on.
IndexKey	**Variant**	In	An expression that specifies the position of a member of the collection. If a numeric expression, *IndexKey* must be a number from 1 to the value of the collection's **Count** property.

8.14.12.3 Exists

Description: Checks if a specific StepBound object exists in the collection by position.

Return type: A **Boolean** value. True if the object exists, else False.

Syntax: *object*.**Exists**(*IndexKey*)

Obj./Arg.	Type	Dir	Description
object	**StepBounds**		The StepBounds object to operate on.
IndexKey	**Variant**	In	An expression that specifies the position of a member of the collection. If a numeric expression, *IndexKey* must be a number from 1 to the value of the collection's **Count** property.

8.14.13 StepBound object

The **StepBound** object contains start and stop times for one step or zone in the trace.

8.14.13.1 Properties

The following table lists all properties on the **StepBound** object:

Property Name	Return Type	Description
StepNo	**Long**	The step number that the start and stop times apply to.
StartTime	**Single**	The time difference from Ts to when the step started.
StopTime	**Single**	The time difference from Ts to when the step started.

The syntax for accessing the properties of the **StepBond** object is as follows:

object.**PropertyName**

where *object* is an expression that evaluates to an object of type **StepBound**.

8.14.14 How to use the API

The API is developed in Visual Basic 6 as an ActiveX object and is as such callable from other languages as well. Below are some examples on how to call the API from Visual basic.

Environment

Set up your Visual Basic environment according to the following

- PowerMACSAPI must be installed on the computer
- PowerMACSAPI must be checked in the project references dialog.
- Do not create more than instance of the PowerMACSAPI.Api object in your application.

Declare a variable at module level

```
Private mPowApi As PowerMACSApi.Api
```

Put some initiating code in the form load event

```
Set mPowAPi = New PowerMACSApi.APi
mPowApi.IpAddress = ”192.168.0.1”
```

Reading cycle data from the target system

This example assumes that there is a label where you can put the cycle data result if there is one. Normally you would loop and save the cycle data to disc or showing them continuously. Here we just do one call to see if there are any cycle data waiting for us.

```
Private Sub cmdGetCd_Click()
'
'  Abstract:  Get cycle data from API-server and display it
'
   On Error GoTo ErrorHandler
   Dim Size As Long
   Dim Ret As RetCodeEnum
   Dim Buf As String

   Size = 10000
   Ret = mPowApi.GetCycleData(Buf, Size)
   If Ret = eRetOk Then
      lblResult = Buf
   ElseIf Ret <> eRetNoData Then
      MsgBox "GetCycleData returned:  " & Ret
   End if
   Exit Sub

ErrorHandler:
   libCleanTerminate "cmdGetCd_Click"
   Stop
   Resume
End Sub
```

Setting a Boolean value in the target system

This example assumes that there are textboxes for address and value, and a dropdown list for the bit value.

```
Private Sub cmdSetBool_Click()
'
'  Abstract:  Set PLC Bool from API-server and display it
'
   On Error GoTo ErrorHandler
   Dim Ret As RetCodeEnum
   Dim Address As Integer
   Dim Value As Boolean

   If Val(txtAddress) < 1 Or Val(txtAddress) > 999 Then
      MsgBox "Address must be between 1 and 999"
      txtAddress.SetFocus
      Exit Sub
   End If

   Address = CInt(txtAddress)
   Value = CBool(txtValue)

   Ret = mPowApi.SetPLCBool(Address, Val(cboBit), Value)
   if Ret <> eRetOk then msgBox "SetPLCBool returned:  " & Ret
Exit Sub

ErrorHandler:
   If Err.Number = 13 Then
      MsgBox "Value must be a valid boolean"
      txtAddress.SetFocus
      Exit Sub
   End If
   MsgBox err.description
   exit sub
End Sub
```

Reading an Integer PLC value from the target system

This example assumes that there are textboxes for address and value.

```
Private Sub cmdSetBool_Click()
'
'  Abstract:  Get PLC Integer from API-server and display it
'
   On Error GoTo ErrorHandler
   Dim Ret As RetCodeEnum
   Dim Address As Integer
   Dim Value As Boolean

   If Val(txtAddress) < 1 Or Val(txtAddress) > 999 Then
      MsgBox "Address must be between 1 and 999"
      txtAddress.SetFocus
      Exit Sub
   End If

   Address = CInt(txtAddress)
   Ret = mPowApi.GetPLCInt(Address, Value)
   if Ret <> eRetOk Then
      msgBox "GetPLCInt returned:  " & Ret
   else
      txtValue = CStr(Value)
   end if
   Exit Sub

ErrorHandler:
   MsgBox err.description
   exit sub
End Sub
```

8.15 GM DeviceNet

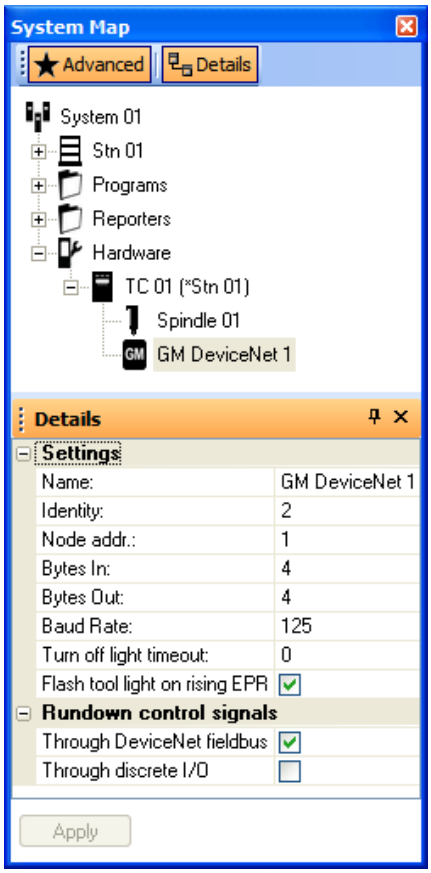
This device type is used to interface a PowerMACS station to an overriding control system according to GM's specification the General Motors document "GAE Electric Nutrunner Controller Design Criteria", GAEc-03 6-9-03Revision.doc. Using it the control system start and stop the tightening cycles and receive result.

The device offers the following two interfaces:

- A special configuration of the standard DeviceNet fieldbus (the input and output maps are predefined and fixed).
- A discrete I/O using digital signals.

By default the discrete I/O interface is mapped to PowerMACS local I/O system (see chapter: I/O) but is also available in the PowerMACS PLC and can therefore be mapped to other kind of interfaces if so wanted. See chapter: GM DeviceNet variables for a description of the PLC interface.

The device is configured using the following form:



Use **Name** to specify the name of the device.

Enter the **Node addr.** (can be [0..63]) and **Baud Rate** (can be 125, 250 or 500 kBit/s).

Specify in **Bytes In** the number of input bytes, this could be either 4 or 8 depending on if the PVI number should be read from the fieldbus or not.

Depending on the number of spindles in the system set **Bytes Out** to 4 for a station with 1 to 5 spindles, and 8 for a station with 6 to16 spindles.

Specify in **Turn of light timeout** how many seconds the external tool lights should present the result of the rundown (max 60 seconds). Specify 0 if the tool lights should remain on to the next rundown cycle.

The checkbox **Flash tool light on rising EPR signal** controls whether or not the green tool lights should start to flash when a positive edge is detected on the Error Proofing Ready input signal. If checked the tools will flash.

Use the checkboxes in the **Rundown control signals** frame to select which interface that should be active. It is possible to have both interfaces active at the same time. If both are active then the corresponding input signals will be OR:ed together.

Note 1: To ensure Operator Safety it is possible to disable all of the input bits from the DeviceNet interface as well as the discrete I/O interface. Uncheck the checkbox in the **Rundown control signals** frame that corresponds to the interface you want to disable.

It is also possible to disable all DeviceNet inputs except for the PVI number using the PLC output GM_IN.DISABLEDNETINP (normally mapped to the digital input DI_GM_DisableDNetInp). This signal is read by the GM Device regardless of the value of the **Through discrete I/O**. See also chapter: GM DeviceNet variables.

Note 2: When using this device type you must ensure that the ANYBUS_in and ANYBUS_out drivers of the PowerMACS PLC are disabled. This is done using the I/O Configuration dialogue opened by double-clicking on the **IO_Configuration** worksheet of the Station that has the fieldbus interface mounted. See chapter: Access to PLC data for details.

Also, since the interface layout is fixed it is not possible to create a reporter for a device of this type.

For more information on the DeviceNet fieldbus, for example the meaning of the status indication LED's, see chapter: Fieldbus Interface.

8.15.1 Interface definition input

The input map definition for signals to PowerMACS is according to the following table:

Word	Byte	Bit	Name in DeviceNet map	Name of IO signal	Pass through IO signal
0	0	0	Green Light (Pass through to IO)		DO_GM_GreenLight
		1	Yellow Light (Pass through to IO)		DO_GM_YellowLight
		2	Red Light (Pass through to IO)		DO_GM_RedLight
		3	Alarm Horn (Pass through to IO)		DO_GM_AlarmHorn
		4-6	Spindle – Mode Set Code (1-8)	DI_GM_ParSet1 - 4	DO_GM_ParSet1 - 4
		7	Spare		
	1	0	Start Cycle	DI_GM_Start	
		1	Reverse	DI_GM_Reverse	
		2	Error Proofing Ready (EPR)	DI_GM_EprIn	DO_GM_EprOut
		3	Spindle – Stop/Disable	DI_GM_StopDisable	
		4-7	Spare		
1	0-1	N/A	Spare		
2-3	0-3	N/A	PVI number		

The inputs are used as follows (all signals have positive logic where active is 1 and passive is 0):

Green Light: This signal is just passed through to PowerMACS discrete I/O interface (local I/O).

Yellow Light: This signal is just passed through to PowerMACS discrete I/O interface (local I/O).

Red Light: This signal is just passed through to PowerMACS discrete I/O interface (local I/O).

Alarm Horn: This signal is just passed through to PowerMACS discrete I/O interface (local I/O).

Spindle – Mode Set Code: Used to choose the mode that should be run when **Start Cycle** goes high (could be 0 to 7 witch will correspond to Mode 1 to 8 in the stations Mode Table). This signal is sampled continuously as long as no cycle is running and is echoed to the output signal **Spindle – Mode Set Code**.

Reverse: When this signal is 1 it will override the value of **Mode Set Code** and force mode to the value 9 for a reverse program. This signal is sampled continuously as long as no cycle is running.

Start Cycle: Starts the tightening cycle if the **Error Proofing Ready** signal has switch from 0 to 1 since the last cycle and **Spindle - Stop/Disable** is low. The cycle is started using the current value of **Spindle – Mode Set Code** or **Reverse**. Immediately when starting the cycle all green, yellow and read indication lamps are turned off (for the DeviceNet and as well as the I/O interface).

Error Proofing Ready: This signal indicates that a new cycle can begin and enables the station to start a cycle. It is also passed through to the output **Spindle Error proofing Ready (EPR),** see Interface definition output. If enabled by the parameter **Flash tool light on rising EPR signal** (see chapter: GM DeviceNet) the green tool lights will start to flash when this signal changes state from low to high. The lights will flash with an interval of 750 ms until the cycle is completed and the result is presented.

Spindle Stop/Disable: Is used to stop the current cycle and to inactivate the spindles from any future cycle. Must be 0 to permit a new cycle to start.

PVI number: The value of this input is added to the cycle data produced when running a cycle as the station level result variable "Wp ID". Sampled when the **Start Cycle** changes from 0 to 1.

DI_GM_DisableDNetInp: This input is only available via the discrete I/O. When set to 1 all inputs via the DeviceNet interface, except for the PVI number, are disabled.

Note! This input is always read, regardless of the value of the device parameter **Through discrete I/O** (see chapter: GM DeviceNet).

8.15.2 Interface definition output

Two output maps are defined in the PowerMACS, one for systems with up to five (5) spindles and one for systems with up to sixteen (16) spindles. Only one of the maps can be enabled at a given time. The maps are enabled/disabled manually by changing the **PLC bytes Out** parameter. All outputs are stable for at least 500 ms. The output map definition for signals from PowerMACS is according to the following table:

Word	Byte	Bit	Name in DeviceNet map	Name of I/O signal
0	0	0	Spare	
		1	Spare	
		2-3	Spare	
		4-6	Spindle – Mode Set Code (0-7)	DO_GM_ParSet1 - 4
		7	Global Accept	DO_GM_GlobalAccept
	1	0	Spindle – In Cycle	DO_GM_InCycle
		1	Spare	
		2	Fault Relay	DO_GM_FaultRelay
		3	Spindle – Cycle Complete	DO_GM_CycleComp
		4	Spindle – Error Proofing Ready (EPR)	DO_GM_EprOut
		5-7	Spare	
1	0	0	Spindle – 1 Good Rundown (Green Light)	DO_GM_Green_Sp1
		1	Spindle – 1 Remove (Red Light)	DO_GM_Red_Sp1
		2	Spindle – 1 Low Torque (Yellow Light)	DO_GM_Yellow_Sp1
		3	Spindle – 2 Good Rundown (Green Light)	DO_GM_Green_Sp2
		4	Spindle – 2 Remove (Red Light)	DO_GM_Red_Sp2
		5	Spindle – 2 Low Torque (Yellow Light)	DO_GM_Yellow_Sp2
		6	Spindle – 3 Good Rundown (Green Light)	DO_GM_Green_Sp3
		7	Spindle – 3 Remove (Red Light)	DO_GM_Red_Sp3
	1	0	Spindle – 3 Low Torque (Yellow Light)	DO_GM_Yellow_Sp3
		1	Spindle – 4 Good Rundown (Green Light)	DO_GM_Green_Sp4
		2	Spindle – 4 Remove (Red Light)	DO_GM_Red_Sp4
		3	Spindle – 4 Low Torque (Yellow Light)	DO_GM_Yellow_Sp4
		4	Spindle – 5 Good Rundown (Green Light)	DO_GM_Green_Sp5
		5	Spindle – 5 Remove (Red Light)	DO_GM_Red_Sp5
		6	Spindle – 5 Low Torque (Yellow Light)	DO_GM_Yellow_Sp5
		7	Spare	

Note! The above shows the output for a system with up to 5 spindles. For a system with up to 16 spindles just continue the table with spindle 6, and so on.

The outputs are set as follows:

Spindle – Mode Set Code: This signal echoes back the current value of the input **Spindle – Mode Set Code** as long as the station is idle, that is, while the output **Spindle – In Cycle is 0**.

Global Accept: This signal is set to 0 when a cycle starts. When the cycle is finished it is set according to the resulting status of the station. It is set to 1 if the cycle completed successfully and 0 if not.

Spindle – In Cycle: This signal is set to 1 when the station is running a cycle, and it is set to 0 when the cycle is finish.

Fault Relay: This signal is currently not used. Always 0.

Spindle – Cycle Complete: This signal is set to 1 when the station has finished a cycle. It is set to 0 when a new cycle starts

Spindle – Error Proofing Ready: This signal is passed through from the input, see Interface definition input.

Spindle – x Good Rundown: See Indication schema below for a description of how this output is set.

Spindle – x Remove: See Indication schema below for a description of how this output is set.

Spindle – x Low Torque: See Indication schema below for a description of how this output is set.

Note! All Outputs have a 500-millisec transition from ON/OFF states to enable sufficient dwell time for PLC's to scan/read the change of state of all Inputs and Outputs.

8.15.2.1 Indication schema

Indication of the bolt result is done differently depending on if the strategy used for the tightening.

Whether or not a tightening is classified as using Torque or Angle control is based on the type of the last step in the main part of the tightening program used, that is, the step before the first CE step in the program.

However, some steps are not classified to be of Torque or Angle control type. A typical example of this is the SR – Socket Release step. To ignore these steps when deciding the control strategy of the program the program is traversed backward, starting at the last step, until one of the steps in the below table is found.

The first found step defines the control strategy of the program.

Step type	Defines Control Strategy as
DT - Run until Dyna-TorkTM DT2 - Run until Dyna-TorkTM, method 2 DT3 - Run until Dyna-TorkTM, method 3 T - Run until Torque TC - Run until Torque with Current Control	**Torque Control.**
AO - Run until Angle with overshoot compensation A - Run until Angle PA - Run until Projected angle TA – Run Torque-Angle with no stop	**Angle Control.**
All other step types	**Undefined**

Depending on the outcome of the Monitoring result and the Control strategy of the used program the result for each bolt is indicated according to the following schemas:

Torque Control strategy:

		TORQUE (Bolt T)				
		< ZDR	**Low**	**OK**	**High**	**> Remove limit**
ANGLE (Bolt A)	**Low**	Yellow	Yellow	Yellow + Red	Yellow + Red	Red
	OK		Yellow	Green	Yellow + Red	Red
	High		Red	Red	Red	Red

If a REHIT is detected (see below for how) then the Yellow + Red outputs are set.

Angle Control strategy:

		TORQUE (Bolt T)				
		< ZDR	**Low**	**OK**	**High**	**> Remove limit**
ANGLE (Bolt A)	**Low**	Yellow	Yellow + Red	Yellow + Red	Yellow + Red	Red
	OK		Yellow + Red	Green	Yellow + Red	Red
	High		Red	Red	Red	Red

If a REHIT is detected (see below for how) then the Yellow + Red outputs are set.

Whether torque and angle are considered OK or not for a bolt is decided by the data in the corresponding Bolt level result variables "Errors". The below tables shows which error bits that are used, and for what purpose.

TORQUE status	Bit in "Errors"
< ZDR	TTNOTRM (Threshold torque not reached for angle monitoring, see chapter: Monitoring Check -- Angle)
REHIT	CROSSGR (Cross gradient restriction exceeded, see chapter: Cross Thread and Gradient)
> Remove limit	RFTLIMRE (Remove Fastener Torque Limit REeached, see chapter: Program)
Low	TLM (Final torque too low during monitoring, see chapter: Monitoring Check – Final Peak Torque)
High	THM (Final torque too high during monitoring, see chapterMonitoring Check – Final Peak Torque)
OK	True **only** if the status of the bolt is OK

ANGLE status	Bit in "Errors"
Low	ALM (Angle too low during monitoring, see chapter: Monitoring Check -- Angle)
High	AHM (Angle too high during monitoring, see chapterMonitoring Check -- Angle)
OK	True **only** if the status of the bolt is OK

If several of the above errors occur during a cycle they are prioritised so that the worse, or most "red", error wins. That is, the priority order is (most significant error has lowest number):

1. The **Red** lamp only
2. The **Yellow** + **Red** lamps
3. The **Yellow** lamp
4. The **Green** lamp.

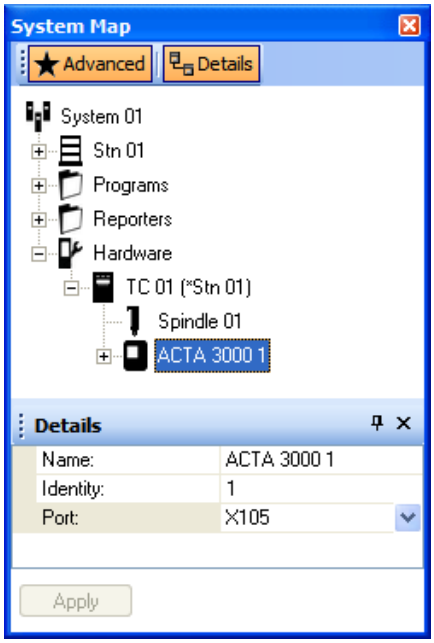
Note! If a bolt ends with status NOK and none of the errors listed above are at hand, or the control strategy is undefined, then none of the lamps will be switched on.

This situation can occur if the monitoring checks and restrictions listed above are not used, or if the program does not include any step defining a control strategy.

The monitoring checks must always be evaluated, thus **Ignore monitoring errors when step ends NOK** cannot be checked.

8.16 ACTA 3000

The ACTA 3000 device enables calibration of the torque transducer for the spindles of your system using the ACTA test equipment from Atlas Copco. One device enables calibration of all spindles in the station. Add the device to the Station TC and make the physical connection of the ACTA unit to this TC.



Use **Name** to specify the name of the device.

Port specifies which physical communication port that should be used. Port 2 is default and should be used for the standard cables.

You can control whether or not cycles can be started from the PowerMACS PLC or the Test Bolts form using the PLC System Globals Station variables ACTA_ACTIVE and DISBALE_ACTA.

The PLC program created by default (when no GM DeviceNet device is used) blocks out starts from the PowerMACS PLC and the Test Bolts form.

The calibration can be run in two different modes, controlled by the ACTA or controlled by the TC.

Controlled by ACTA

If the calibration is controlled by the ACTA the cycles are started from the ACTA unit. In the ACTA unit you have to specify the spindle and the mode number that shall be used. It is only this spindle that will run.

If mode 0 is selected a special program adapted to the QRTT equipment is used for the calibration. With this program both the torque and angles can be verified and calibrated.

If a mode between 1 and 50 is selected the program that is specified for that mode in the mode table is used for the calibration. This can be any tightening program that is included in the setup. In this case there is only possible to calibrate the torque.

Controlled by TC

If the calibration is controlled by the TC the cycles are started from the TC as all normal tightenings. The start can come from a digital I/O, a fieldbus or some other external equipment. It is still possible to select the mode number from the ACTA if wanted. The PLC system global ACTAMODE will reflect the mode selected from the ACTA and it is possible to connect this to the output MODE.

Calculation of Torque Scale Factor

The ACTA unit can calculate a new torque scale factor, based on the torque measurements that were made in the TC and in the ACTA. It is also possible to save this value in the TC if the difference from the old scale factor is less than 10%.

Requirements on the tightening program used for calibration

Any normal tightening program can be used for the calibration, but it is important to consider how the ACTA detects cycle start and cycle end to make sure the values sent from the TC really correspond to the values measured in the ACTA

See Atlas Copco Tools document No. 9836 2086 00 [4] for how to operate the ACTA test equipment.

8.17 Process data

Process data comprises following types of data:

- Cycle data
- Traces
- Events
- Setup

Process data is accessed from external devices using fieldbus, external communication protocols, the API, etc.

Cycle data, Events, and Traces generated by PowerMACS are put into a number of FIFO queues. Each device has their own queues, one for cycle data, one for traces, and one for events. This means that they can read data at their own speed without interfering with each other.

All access to Process data is done on the client's request. When requested for, the PowerMACS system will return the next data from the queue in question. Should the queue be empty then the client will receive an empty data package. It is the client's responsibility to request new data. PowerMACS will not spontaneously indicate to him that new data is available.

Since a process data item, e.g. one cycle data, often are bigger than what can be handled by a single read or write access, it is accessed through successive reads or writes where each access returns a part of the complete data item. These parts are called packages. In other words, one data item consists of one or more data packages.

A Process data package has the following format:

Byte	
0	SEQ (Most Significant Byte)
1	SEQ (Least Significant Byte)
2	LEN (Most Significant Byte)
3	LEN (Least Significant Byte)
4	Data byte 1
5	Data byte 2
:	:
N + 4	Data byte N

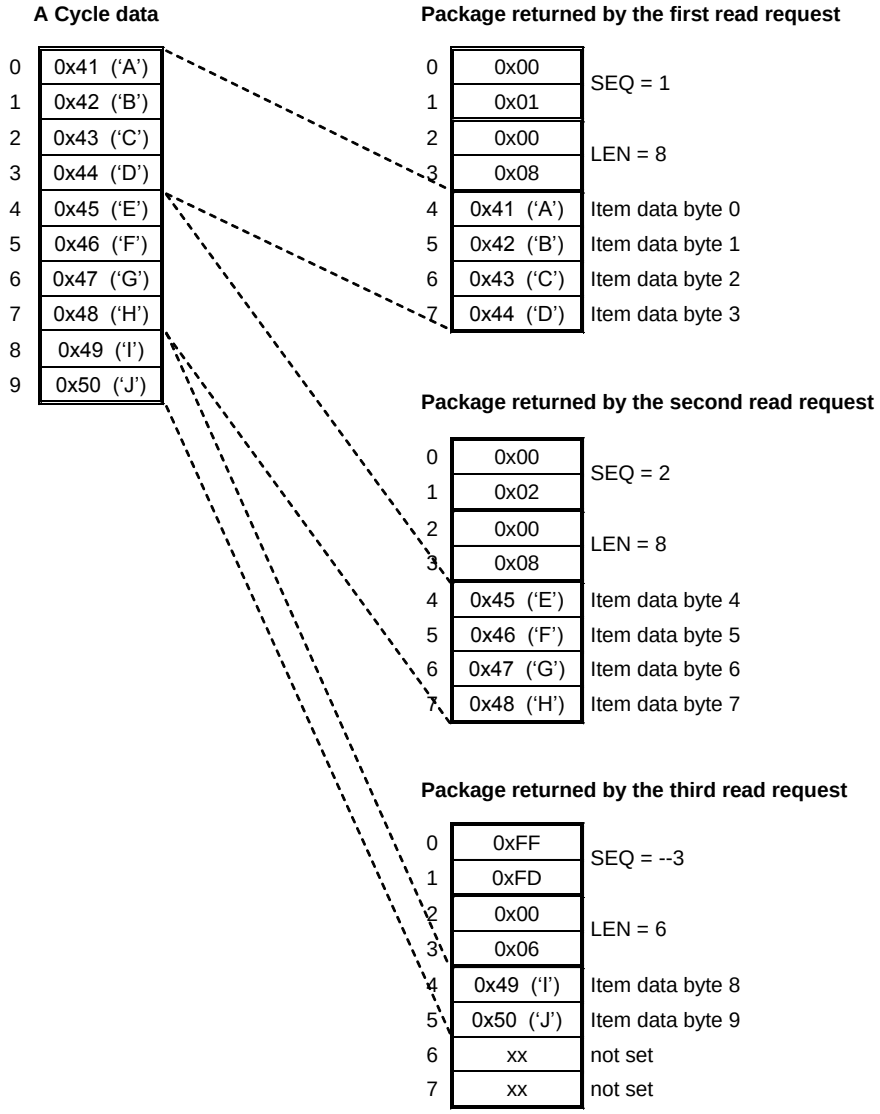
First in the package there is a sequence number named **SEQ** and a length field named **LEN**. Both are Short Integers, which means that they occupy two bytes each (see Data types).

SEQ is the sequence number of the package within a particular data item (e.g. a cycle data). The first package has **SEQ** = 1, the second has **SEQ** = 2, and so on. To indicate the last package of a data item **SEQ** is negated. This means that a data item that is divided in three packages will have the following values of **SEQ**: 1, 2, -3.

LEN is the length of the complete package, including both **SEQ** and **LEN**. **LEN** is therefore always equal to N + 4 where N is the number of data bytes contained in the package.

To reconstruct the Data item being read you just have to concatenate the Data part (Data byte 1, …) of each package, from first package (**SEQ** = 1) to the last package (**SEQ** = -K).

Example: Assuming that a complete cycle data consists of the 10-character string "ABCEDFGHIJ" should be read using an 8-byte buffer then the below data transfer will occur when the cycle data is requested (all data is written in Hexadecimal notation):



If this was the last available cycle data in the queue then another read would give the following result (indicating that the queue is empty):

Empty package returned since queue is empty

0	0xFF	SEQ = --1
1	0xFF	
2	0x00	LEN = 4
3	0x04	
4	xx	not set
5	xx	not set
6	xx	not set
7	xx	not set

8.17.1 Data types

The following data types are used in the following sections:

Data type	Description	Values
Boolean	2 bytes, 16 bit unsigned binary	Zero (0) represent False and non-zero True
Short integer	2 bytes, 16 bit, 2-complement signed binary	-32768, …, +32767
Long integer	4 bytes, 32 bit, 2-complement signed binary	-2 147 483 648, …, +2 147 483 647
Time	4 bytes, 32 bit long integer	Number of seconds since January 1, 1970
Float or Real	4 bytes, 32 bit, IEEE-754 standard	+/- 1.18 * 10^{38}
char[]	Array of bytes containing ASCII-character	Null-terminated i.e. the byte after last relevant character contains null, 0.

For data occupying more than one byte the order is that **most** significant byte comes first, at the lowest address. This is also known as Big Endian or Motorola format.

8.17.2 Layout of Cycle data in Process data

The layout of Cycle data is set up by use of a Reporter.

8.17.3 Layout of Events in Process data

An Event contains data as described in the table below.

Offset	Item	Data type	Description
0	DataNo	Short integer	A device unique sequence number. This number is incremented by one for each event reported over the device.
2	SeqNo	Long integer	A global sequence number. This number is incremented by one for each event generated by the system.
6	Time	Time	Date and time for event.
10	Code	Short integer	Event code, see List of events for possible values.
12	Type	Short integer	Event type, see View Event Log for possible values.
14	Station	Short integer	Station on which the event occurred, 0 if not relevant
16	Bolt	Short integer	Bolt for which the event occurred, 0 if not relevant
18	Observed	Short integer	<> 0 if event is observed
20	Pgm	char[21]	Tightening program connected to the event, 0 if not relevant
41	EventStr	char[81]	Event string in the language configured using ToolsTalk PowerMACS.
122	Severity	Short integer	The severity of the event, see View Event Log for possible values.
124	Par 1	char[4]	First parameter for the EventStr, see below for a description.
128	Par 2	char[4]	Second parameter for the EventStr, see below for a description.
132	Par 3	char[4]	Third parameter for the EventStr, see below for a description.
136	Par 4	char[4]	Fourth parameter for the EventStr, see below for a description.
140	Par 5	char[4]	Fifth parameter for the EventStr, see below for a description

Par1 to Par5 represents values that should replace the parameter placeholders in the event strings defined for each event code (%d, %x, %f, and %c). See List of events for a listing of all events with their place holders. Par1 represents the leftmost placeholder, Par2 the second leftmost, and so on. The respective placeholder has the following meaning:

Placeholder	Meaning
%d	Interpret the four bytes of ParX as a long and display it as a decimal integer
%x	Interpret the four bytes of ParX as a long and display it as a hexadecimal integer
%f	Interpret the four bytes of ParX as a float and display it as a float with two (2) decimals
%c	Interpret the four bytes of ParX as ASCII values of four single characters and display them as characters.

Which items that are included for an event when it is read by a particular device is controlled by the Reporter connected to the device.

If all items are included then the size of an event is 144 bytes.

8.17.4 Layout of Traces in Process data

A Trace contains data as described in the table below.

Offset	Item	Data type	Description
0	DataNo	Short integer	A device unique sequence number. This number is incremented by one for each trace reported over the device.
2	FmtVer	Long integer	Format version on the TC when the trace was created
6	Station	Long integer	Station
10	Bolt	Long integer	Bolt
14	Time	Time	Timestamp of trace
18	Freq	Float	Sampling frequency in Hz
22	TimeOffset	Long integer	Offset to first sample in milliseconds
26	NoTsamples	Long integer	Number of torque samples [0..2000]
30	NoAsamples	Long integer	Number of angle samples [0..2000]
34	NoCsamples	Long integer	Number of current samples [0..2000]
38	Not used	Long integer	Always zero (0)
42	PgmName	char[21]	Name of program used at the occasion
63	1st T sample	Float	The first Torque sample
63 + 4 * (NoTsamples – 1)	Last T sample	Float	The last Torque Sample (sample NoTsamples)
63 + 4 * (NoTsamples)	1st A sample	Float	The first Angle sample
63 + 4 * (NoTsamples + NoAsamples – 1)	Last A sample	Float	The last Angle Sample (sample NoAsamples)
63 + 4 * (NoTsamples + NoAsamples)	1st C sample	Float	The first Current sample
63 + 4 * (NoTsamples + NoAsamples + NoCsamples – 1)	Last C sample	Float	The last Current Sample (sample NoCsamples)

The trace will only contain the samples that are recorded. This means that only data for used channels are recorded. The number of samples included for each channel is given by the variable NoTsamples, NoAsamples, NoCsamples and NoGsamples respectively.

8.17.5 Layout of Setup Item Descriptions

Using the API interface it is possible to read or write individual items within a setup. Items that can be accessed are parameters for which alteration would be meaningful and which can be handled by the system.

The following parts of a setup may be modified individually:

- Bolt
- Spindle – Parameters
- Spindle – Channel data
- Programs – General data
- Programs – Monitoring data
- Sequences and programs – Step Control data
- Sequences and programs – Step Speed data
- Sequences and programs – Step Speed ramp down data
- Sequences and programs – Step Surveillance data

To access an individual item a description must be provided which points to the item. This description should be according to the syntax described in the following.

8.17.5.1 Bolt

Syntax: **Bolt[**<station no>,<bolt no>**].**<parameter>

<station no> = 1..Max station no
<bolt no> = 1..Max bolt no in that station
<parameter> = See table below

<parameter>	**Data type**	**Description**
OpMode	Short integer	Operation mode: • 0: Connected • 1: Disconnected OK • 2: Disconnected NOK
ExternalBoltNo	Short integer	The bolt number used in cycle data reports.

Example: `Bolt[1,1].OpMode`

8.17.5.2 Spindle, Parameters

Syntax: **`Spindle[`**`<spindle no>`**`].`**`<parameter>`

`<spindle no>` = 1..Max spindle no
`<parameter>` = See table below (and Spindle Set Up for a description)

<parameter>	Data type	Description
NoCyclMaint	Long integer	Service interval in no. of cycles (>= 0)
Direction	Short integer	Direction in which the spindle should rotate. Allowed values are: 0. Forward (clockwise) 1. Backward (counter clockwise)
ZeroSpeed	Float	The maximum speed that counts as zero speed
ADiffEnabled	Boolean	If angle channel difference test should be enabled
ADiffUseSpindle	Boolean	If spindle's limits should be used
ADiffMaxDiff	Float	Maximum allowed difference
ADiffSamples	Short integer	Maximum number of samples tolerated above ADiffMaxDiff
TDiffEnabled	Boolean	If torque channel difference test should be enabled
TDiffUseSpindle	Boolean	If spindle's limits should be used
TDiffChannels	Short integer	Channels to compare 1 : Current compared to Torque Channel 1 2 : Current compared to Torque Channel 2 *(3 : Torque Channel 1 compared to Torque Channel 2 (* *) Requires two torque channels
TDiffMaxDiff	Float	Maximum allowed difference
TDiffSamples	Short integer	Maximum number of samples tolerated above TDiffMaxDiff
ContFailedFzo	Boolean	True if spindle should continue to run after failed flying zero offset check.
SzoMax1	Float	Max allowed static zero offset for transducer 1.
SzoMax2	Float	Max allowed static zero offset for transducer 2.
DzoMax1	Float	Max allowed dynamic zero offset for Torque 1.
DzoMax2	Float	Max allowed dynamic zero offset for Torque 2.
FzoMax1	Float	Max allowed flying zero offset for Torque 1.
FzoMax2	Float	Max allowed flying zero offset for Torque 2.
ExtEnabled	Boolean	True when external equipment enabled.

<parameter>	Data type	Description
ExtInverse	Boolean	True if external equipment inverses direction.
ExtGearRatio	Float	External Equipment GearRatio
ExtTorqueLossF	Float	Torque Measurement Loss factor for transducers, < 1.0
ExtWindUp1	Float	External Equipment Replacement WindUp for Angle Channel 1.
ExtWindUp2	Float	External Equipment Replacement WindUp for Angle Channel 2.

Example: `Spindle[1].Direction`

8.17.5.3 Spindle, Channel data

Syntax: **`Spindle[`**`<spindle no>`**`].Chan[`**`<channel no>`**`].`**`<parameter>`

`<spindle no>` = 1..Max spindle no
`<channel no>` = Channel number, 1..6 (number is given by the order in the form)
`<parameter>` = See table below (and Spindle Set Up for a description)

<parameter>	Data type	Description
Filter	Short integer	Filter frequency (for the torque and current channels)

Example: `Spindle[1].Chan[1].Filter`

8.17.5.4 Program, General data

Syntax: **Program[**<pgm name>**].General.**<parameter>

<pgm name> = The name of the program
<parameter> = See table below description)

<parameter>	Data type	Description
StopMonAtNokStep	Boolean	If True then monitoring is stopped when a step have a NOK status when it ends.
Use2ndMonBuf	Boolean	If true then 2nd monitoring buffer is used if there is more than one used torque or angle channel.
ReportTqLimit	Float	Report threshold torque limit. If none of the bolts executed for a cycle (disconnected bolts excluded) exceeded this value then the corresponding cycle data will be automatically dropped
RemFastTqLimit	Float	Remove fastener torque limit. When the measured torque exceeds the entered value the current step is immediately stopped.

Example: `Program[Pgm01].General.StopMonAtNokStep`

8.17.5.5 Program, Trace data

Syntax: **Program[**<pgm name>**].Trace.**<parameter>

<pgm name> = The name of the program
<parameter> = See table below description)

<parameter>	Data type	Description
TraceStart	Short integer	The main condition for when to start recording a Trace. Takes the following values: • 1: At cycle start • 3: When the step with number <TraceStartStep> is started.
TraceStartStep	Short integer	The number of the step where trace will start
TraceStartQty	Short integer	Specifies an addition condition for when to start recording a trace. Takes one of the following values: • 0: None • 1: When the measured angle reaches the value <TraceStartMin> • 2: When the measured torque reaches the value <TraceStartMin> 3: When the measured current reaches the value <TraceStartMin>
TraceStartMin	Float	The minimum value to reach before trace starts. Valid only if <TraceStartQty> has the value 1, 2 or 3.
Inc2ndTorque	Boolean	If true both torque channels will be recorded to trace.
Inc2ndAngle	Boolean	If true both angle channels will be recorded to trace.

Example: `Program[Pgm01].General.StopMonAtNokStep`

8.17.5.6 Program, Monitoring data

Syntax: **`Program[`**`<pgm name>`**`].Mon.`**`<parameter>`

`<pgm name>` = The name of the program
`<parameter>` = See table below (and Bolt Monitoring for a description)

<parameter>	Data type	Description
MonStart	Short integer	Defines when to start monitoring. Takes the following values: • 0: Disabled • 1: At cycle start • 3: When a step that has the Start/restart monitoring flag set is started (see Step – Control). • 4: When the measured torque reaches the value <MonStartTorqueVal>.
MonEnd	Short integer	Defines when to end monitoring. Takes the following values: • 1: At cycle end • 3: When the measured torque reaches the value <MonEndTorqueVal>.
MonStartTorqueVal	Float	The torque value to reach before starting monitoring. Only valid if <MonStart> is 4.
MonEndTorqueVal	Float	The torque value to reach before ending monitoring. Only valid if <MonEndt> is 3.
MonBufAutoScale	Boolean	If True then the resolution of the monitoring buffer is auto scaled.
MonResolution	Short integer	No of angle increment per buffer element in the monitoring buffer. Valid only if <MonBufAutoScale> is False.
MonBufOverrunAllowed	Boolean	If True then monitoring buffer overrun is not regarded as an error. Valid only if <MonBufAutoScale> is False.

Example: `Program[Pgm01].Mon.MonStart`

8.17.5.7 Sequence and Program, Step, Control data

Syntax: **Program[**<pgm name>**].Step[**<step no>**].Control.**<parameter>
Sequence[<seq name>**].Step[**<step no>**]. Control.**<parameter>

<pgm name> = The name of the program
<step no> = Step number, 1..50
<parameter> = See table below (and Step – Control for a description)

<parameter>	**Data type**	**Description**
Sync	Boolean	If True then the step is synchronized.
StartMon	Boolean	If True then monitoring will be cleared and restarted when the step is started.
Par[<par no>]	Float	The control function parameter (allowed values are 1..7). The number of each parameter is given by the order they are listed in chapter: Step – Control.

Example:
```
Program[Pgm01].Step[1].Control.Sync
Program[Pgm01].Step[1].Control.Par[1]
```

8.17.5.8 Sequence and Program, Step, Speed data

Syntax: **Program[**<pgm name>**].Step[**<step no>**].Speed.**<parameter>
Sequence[<seq name>**].Step[**<step no>**].Speed.**<parameter>

<pgm name> = The name of the program
<step no> = Step number, 1..50
<parameter> = See table below (and Step – for a description)

<parameter>	**Data type**	**Description**
Speed	Float	Target Speed
Zsd	Boolean	If True then Zero Speed detection is active
Sru	Float	Speed ramp up
Tru	Float	Torque ramp up
RampType	Short integer	Ramp type. Possible values: Straight / Smoothened 1. Straight 2. Smoothened
NoSrds	Short integer	No of Speed ramp downs case used, [0..5]. See Sequence and Program, Step, Speed, Ramp down data.

Example:
```
Program[Pgm01].Step[1].Speed.Speed
```

8.17.5.9 Sequence and Program, Step, Speed, Ramp down data

Syntax: **Program[**<pgm name>**].Step[**<step no>**].Speed.RampDown[**<ramp>**].**<parameter>
Sequence[<seq name>**].Step[**<step no>**].Speed.RampDown[**<ramp>**].**<parameter>

<pgm name> = The name of the program
<step no> = Step number, 1..50
<ramp> = Ramp number, 1..5
<parameter> = See table below (and Step – for a description)

<parameter>	Data type	Description
Start	Short integer	Defines the quantity to monitor as start condition. Allowed values: 1. Torque 2. Angle 3. Current 4. Time
Level	Float	The trig level of the measured quantity.
Speed	Float	The new target speed.
Ramp	Float	The ramp to use when changing the speed.

Example: `Program[Pgm01].Step[1].Speed.RampDown[1].Start`

8.17.5.10 Sequence and Program, Step, Other data

Syntax: **Program[**<pgm name>**].Step[**<step no>**].Other.**<parameter>
Sequence[<seq name>**].Step[**<step no>**].Other.**<parameter>

`<pgm name>` = The name of the program
`<seq name>` = The name of the sequence
`<step no>` = Step number, 1..50
`<parameter>` = See table below

<parameter>	**Data type**	**Description**
StartDelay	Float	Start delay of step
TorqueSpikeAngle	Float	Torque spike elimination angle
StopMethod	Short integer	How to stop. Allowed values: 1. Inertia break 2. Ramp down current 3. Hold Torque
InitialStopValue	Float	The start level of the amperage ramp expressed in % of the amperage at shut off. Used only if <StopMethod> is "Ramp down current".
DownRamp	Float	The amperage down ramp. Used only if <StopMethod> is "Ramp down current".
HoldWhenStopped	Boolean	If True then the position should be maintained when stopped. Can only be used if <StopMethod> is "Inertia break" or "Ramp down current".

Example: `Program[Pgm01].Step[1].Other.StartDelay`

8.17.5.11 Sequence and Program, Step, Surveillance data

Syntax: **Program[**<pgm name>**].Step[**<step no>**].Surveillance [**<id no>**].Par[**<par no>**]**
Sequence[<seq name>**].Step[**<step no>**].Surveillance [**<id no>**].Par[**<par no>**]**

<pgm name> = The name of the program
<step no> = Step number, 1..50
<id no> = Identity number of check/restriction/monitoring function.
<par no> = Parameter number, 1..12, as listed in the description.

Example: `Program[Pgm01].Step[1].Surveillance[1].Par[1]`

8.17.6 Layout of Setups

A Setup is a complete package of data containing all information necessary to configure a complete PowerMACS system. It is stored in a proprietary compressed format, safeguarded by Checkums to keep its integrity. The main purpose for handling setups this way is for storage of backup copies, or for exchange of setups between systems. It is not possible with normal software tools to read or write information within a setup.

9 Specification

9.1 Specification

This part is the specification of the system. It describes general performances of a complete system:

Topic	Performance
Max. time from start signal until all spindles are running	50 msec..
Max. time from last spindle stops a step until all are running in the next step:	50 msec. + monitoring time.
Max. time from last spindle stops last step until OK/NOK status on digital output, per spindle and total:	0.1 sec. + monitoring time.
Max. time between cycles	0.5 sec at limited data reporting.
Position accuracy	0.5 degrees.
Sampling frequency	1000 Hz
Stations per system.	1 – 15
Spindles per system	1 – 50
Delays in set values to servo	2 x sampling time
Maximum time for a loop in the PLC program	5
Maximum number of cycle data stored in the system	Depends on number of bolts and variables included.
Traces per spindle. Each trace with 2000 values for angle, torque and current.	Up to 300 (depends on length of trace and curve form)
Max time to change a device	3 min

The Performance values can be seen as a realistic estimate of performance for the finalized system, but the design of the system is focused on reaching the Target values.

Traces will be stored in two circular buffers with equal size, one for OK cycles, the other for NOK cycles. A new trace will overwrite the oldest stored trace, independent of tightening program.

10 Appendix

10.1 List of events

The below table lists the Code, Compact code, Type, Severity and Event string for most events that can be generated by the system. The compact event code is only used when events are reported from an Ethernet Open Protocol device.

Code	Comp Code	Type	Severity	Event string
1	1	Ext. com	Error	Unexpected signal: %d %d %d
2	2	System	Error	DB Get error %d, Proc %d, Case %d
3	3	Hardware	Error	Spurious interrupt %d, %d, %d
4	4	System	Error	Internal error on line %d in class base no %d, ret %d
101	9	Power up	Info	Power on
102	10	Hardware	Error	System: Lost connection to TC %d
103	11	Hardware	Info	System: Established connection to TC %d
104	12	Hardware	Error	System: TC %d is of wrong type
105	13	Hardware	Error	System: Spindle TC %d is configured as system TC
106	442	Hardware	Error	System: TC %d is of wrong version
107	443	Hardware	Error	System: TC %d participates in another system
108	444	Hardware	Error	System: TC %d needs coldstart
201	14	Setup	Error	Easy View parameter %d fails in pos %d
202	15	System	Error	Error when reading/writing ODB image (err=%d)
203	16	System	Error	Error when reading ODB image (ODB seq.=%d, my seq.=%d)
204	431	System	Error	Error when receiving (type=%d, d1=%d, d2=%d)
205	432	System	Error	PLC ZIP file access error (access=%d, info=%d)
501	901	System	Error	Trying to read event that no longer exists (missing %d)
601	18	System	Warning	Start is impossible since there are no active spindles
603	20	System	Error	Bolt %d is connected to invalid spindle %d
604	21	System	Error	Bolt %d is connected to invalid spindle %d
605	22	System	Error	ST: signal from undefined spindle
606	23	System	Error	ST: signal from undefined bolt
607	24	System	Error	ST: Data structures inconsistent
608	25	System	Error	ST: Telegram %d received in wrong state %d from proc %d
609	26	System	Error	Start is impossible since station has no contact with spindle %d
610	27	System	Error	Start is impossible since station has no contact with PLC

Code	Comp Code	Type	Severity	Event string
611	28	System	Error	Start is impossible since station has no contact with Test Bolts
612	29	System	Error	Unexpected Cycle End when running RM
613	30	System	Error	Invalid Bolt number from spindle
614	31	System	Error	ST: DB read error %d for bolt %d
615	32	Emergency stop	Warning	ST: emergency stopped
616	33	Emergency stop	Warning	ST: machine stopped
617	34	System	Error	Start is impossible since stations has no contact with syinit
618	35	Setup	Error	Cycle data too big, bolt %d not stored
619	419	General	Info	Cycle start disabled
620	425	Setup	Error	Start is impossible since bolt %d and %d both uses TC %d
621	434	System	Error	Station failed to reach TC. Bolt is disconnected (OK)
622	435	System	Error	Station failed to reach TC. Bolt is disconnected (NOK)
623	436	System	Error	Station starts the cycle even though %d spindles could not be reached
624	437	System	Error	Station failed to init servo. Bolt is disconnected (OK)
625	438	System	Error	Station failed to init servo. Bolt is disconnected (NOK)
626	439	System	Error	Station failed to read which cycle data that is wanted
627	445	System	Error	Start is impossible since there are conflicting TCs in the station
628	477	Setup	Info	Cycle data dropped since no bolt reached report threshold torque limit
629	544	Setup	Error	ST: BoltControl from PLC has bad value (%d) in mode %d
630	545	Setup	Error	ST: Found no program for mode %d in mode table to run for the bolt
631	552	System	Error	Start is impossible since station has no contact with Station service
632	560	General	Warning	PLC BOLTCTRL2 cmd %d ignored since no bolts are selected
633	561	General	Error	PLC BOLTCTRL2 cmd %d ignored since station is not idle
634	562	General	Error	PLC BOLTCTRL2 cmd %d ignored since there is no cycle data
635	563	General	Error	PLC BOLTCTRL2 cmd %d ignored since bolt (ordinal no %d) has no cycle data
636	564	General	Info	PLC BOLTCTRL2 cmd %d is ignored since command code is unknown
637	565	General	Info	Cycle data with status %d dropped for bolt (ordinal no %d)
638	566	System	Error	Cycle data does not contain any bolt records
639	889	General	Info	Cycle start is only allowed from Audi XML device

Code	Comp Code	Type	Severity	Event string
640	19	General	Info	Cycle start is not allowed while FILTER_SPINDLEEV signal in PLC is high.
700	36	System	Error	SP: DB read error %d, object %d
701	37	Setup	Error	No current channel configured
703	39	System	Error	SP: Internal error line %d, file %d
706	42	Setup	Error	Incorrect Sampling frequency
708	44	System	Error	Spindle initialization failed
709	45	System	Error	SP: Hunt for ipc process %d failed
711	47	Setup	Error	Incorrect Direction
712	48	Setup	Error	Step %d, Incorrect Torque spike elimination setup
713	49	Setup	Error	Step %d, Incorrect Pre step time
714	50	Setup	Error	Step %d, Incorrect Post step time
720	56	Check	Error	Step %d, Monitoring: Final torque. %f > %f
721	57	Check	Error	Step %d, Monitoring: Threshold angle. %f > %f
722	58	Check	Error	Step %d, Monitoring: Torque rate in interval 1. %.4f > %.4f
723	59	Check	Error	Step %d, Monitoring: Torque rate in interval 1. %.4f < %.4f
724	60	Check	Error	Step %d, Monitoring: Deviation in interval 1. %.4f > %.4f
725	61	Check	Error	Step %d, Monitoring: Yield point torque diff. %f > %f
726	62	Check	Error	Step %d, Monitoring: Yield point angle diff. %f > %f
727	63	Setup	Error	Monitoring: Incorrect configuration
728	64	Setup	Error	Step %d, Monitoring: Incorrect Final torque
729	65	Setup	Error	Step %d, Monitoring: Incorrect Threshold angle
730	66	Setup	Error	Step %d, Monitoring: Incorrect Torque rate deviation (interval %d)
731	67	Setup	Error	Step %d, Monitoring: Incorrect Yield point torque
732	68	Setup	Error	Step %d, Monitoring: Incorrect Yield point angle
733	69	Setup	Error	Trace: Incorrect configuration
734	70	Setup	Error	Step %d, Control: Incorrect Torque
735	71	Setup	Error	Step %d, Control: Incorrect Angle
736	72	Setup	Error	Step %d, Control: Incorrect Time
737	73	Setup	Error	Step %d, Control: Incorrect Current
739	75	Setup	Error	Step %d, Control: Incorrect Percent
740	76	Setup	Error	Step %d, Control: Incorrect No degrees
741	77	Setup	Error	Step %d, Control: Incorrect No of points (NNOS or RNOS)

Code	Comp Code	Type	Severity	Event string
742	78	Setup	Error	Step %d, Control: Incorrect Increment
743	79	Setup	Error	Step %d, Control: Incorrect Direction
744	80	Setup	Error	Step %d, Control: Incorrect Stop Condition
745	81	Setup	Error	Step %d, Control: Incorrect Digital Input
746	82	Setup	Error	Step %d, Control: Incorrect Digital Output
747	83	Setup	Error	Step %d, Control: Incorrect Step type from PLC
748	84	Setup	Error	Step %d, Restriction: Incorrect FS Angle
749	85	Setup	Error	Step %d, Restriction: Incorrect FS Torque
750	86	Setup	Error	Step %d, Restriction: Incorrect FS Time
752	88	Setup	Error	Step %d, Restriction: Incorrect Cross thread and gradient
753	89	Setup	Error	Step %d, Restriction: Incorrect Torque profile
754	90	Check	Error	Step %d, Check: Peak torque. %f > %f
755	91	Check	Error	Step %d, Check: Torque in angle window. %f > %f
756	92	Check	Error	Step %d, Check: Torque in time window. %f > %f
757	93	Check	Error	Step %d, Check: Mean torque. %f > %f
758	94	Check	Error	Step %d, Check: Angle. %f > %f
759	95	Check	Error	Step %d, Check: Time. %f > %f
760	96	Check	Error	Step %d, Check: Current as Torque. %f > %f
762	98	Check	Error	Step %d, Check: T/T3. %f > %f
763	99	Check	Error	Step %d, Check: Post view torque high.
764	100	Check	Error	Step %d, Check: Post view torque low.
765	101	Check	Error	Step %d, Diagnostic: Static Zero Offset high on T1. abs(%f) > %f
766	102	Check	Error	Step %d, Diagnostic: Dynamic Zero Offset high on T1. abs(%f) > %f
767	103	Check	Error	Step %d, Diagnostic: Flying Zero Offset high on T1. abs(%f) > %f
768	104	Check	Warning	Step %d, Diagnostic: Flying Zero Offset high warning on T1. abs(%f) > %f
769	105	Check	Error	Step %d, Diagnostic: Angle count high on A1. %f > %f
770	106	Check	Error	Step %d, Diagnostic: Calib. failed on T1. Meas. %f, expected %f
771	107	Setup	Error	Step %d, Diagnostic: Incorrect configuration
772	108	Setup	Error	Step %d, Check: Incorrect Peak torque
773	109	Setup	Error	Step %d, Check: Incorrect Torque in angle window
774	110	Check	Error	Step %d, Check: Torque in angle window failed
775	111	Setup	Error	Step %d, Check: Incorrect Torque in time window

Code	Comp Code	Type	Severity	Event string
776	112	Check	Error	Step %d, Check: Torque in time window failed
777	113	Setup	Error	Step %d, Check: Incorrect Mean torque
778	114	Check	Error	Step %d, Check: Mean torque failed
779	115	Setup	Error	Step %d, Check: Incorrect T/T3
780	116	Check	Error	Step %d, Check: T/T3 failed
781	117	Setup	Error	Step %d, Check: Incorrect Angle
782	118	Check	Error	Step %d, Check: Angle failed
783	119	Setup	Error	Step %d, Check: Incorrect Current
784	120	Setup	Error	Step %d, Check: Incorrect Clamp Load
785	121	Setup	Error	Step %d, Check: Incorrect Time
786	122	Setup	Error	Step %d, Check: Incorrect Post view
787	123	Check	Error	Step %d, Check: Post view Torque failed
788	504	Check	Error	Step %d, Check: Post view Torque failed (Buffer overflow)
793	128	General	Warning	Spindle %d: Time for service. %d cycles since service (%d to many)
794	129	Setup	Error	Spindle: No program for mode %d
795	130	Setup	Error	Cycle data too big, step %d not stored
796	131	Check	Error	Step %d, Monitoring: Final torque could not be evaluated
797	132	Check	Error	Step %d, Monitoring: Threshold angle could not be evaluated
798	133	Check	Error	Step %d, Monitoring: Torque rate could not be evaluated (interval %d)
799	134	Check	Error	Step %d, Monitoring: Yield point torque could not be evaluated
800	135	Check	Error	Step %d, Monitoring: Yield point angle could not be evaluated
801	136	Check	Error	Step %d, Check: Peak torque. %f < %f
802	137	Check	Error	Step %d, Check: Torque in angle window. %f < %f
803	138	Check	Error	Step %d, Check: Torque in time window. %f < %f
804	139	Check	Error	Step %d, Check: Mean torque. %f < %f
805	140	Check	Error	Step %d, Check: Angle. %f < %f
806	141	Check	Error	Step %d, Check: Time. %f < %f
807	142	Check	Error	Step %d, Check: Current as Torque. %f < %f
809	144	Check	Error	Step %d, Check: T/T3. %f < %f
810	145	Check	Error	Step %d, Diagnostic: Angle count low on A1. %f < %f
811	146	Check	Error	Step %d, Diagnostic: Angle count high on A2. %f > %f
812	147	Check	Error	Step %d, Diagnostic: Angle count low on A2. %f < %f

Code	Comp Code	Type	Severity	Event string
813	148	Setup	Error	Step %d, Speed is not defined
814	149	Setup	Error	Step %d, Control: Slope has bad configuration
815	150	Setup	Error	Step %d, Check: Wrong step type for Check
816	151	Check	Error	Step %d, Check: Angle at the corner failed
817	152	Check	Error	Step %d, Check: Angle at the corner %f < %f
818	153	Check	Error	Step %d, Check: Angle at the corner %f > %f
819	154	Setup	Error	Step %d, Check: Torque has bad configuration
820	155	Check	Error	Step %d, Check: Torque at the corner failed
821	156	Check	Error	Step %d, Check: Torque at the corner %f < %f
822	157	Check	Error	Step %d, Check: Torque at the corner %f > %f
829	164	Setup	Error	Step %d, Check: Shut Off Torque has bad configuration
830	165	Check	Error	Step %d, Check: Shut Off Torque %f < %f
831	166	Check	Error	Step %d, Check: Shut Off Torque %f > %f
832	167	Check	Error	Step %d, Check: Shut Off Torque failed
833	168	Setup	Error	Step %d, Restriction: Torque-Current has bad configuration
834	169	System	Error	Step %d, Received unexpected SP_RUN_STEPS from station.
835	170	General	Error	Step %d, Run reverse before retry failed (max time was exceeded)
836	171	General	Error	Step %d, Run reverse before retry failed (max torque was exceeded)
837	172	General	Error	Step %d, Run reverse before retry failed
838	173	Setup	Error	Step %d, Spindle %d has incorrect Gear ratio
839	174	Check	Error	Step %d, Monitoring: Final torque. %f < %f
840	175	Check	Error	Step %d, Monitoring: Threshold angle. %f < %f
841	176	Check	Error	Step %d, Monitoring: Torque rate in interval 2. %.4f > %.4f
842	177	Check	Error	Step %d, Monitoring: Torque rate in interval 2. %.4f < %.4f
843	178	Check	Error	Step %d, Monitoring: Deviation in interval 2. %.4f > %.4f
844	179	Check	Error	Step %d, Monitoring: Yield point torque diff. %f < %f
845	180	Check	Error	Step %d, Monitoring: Yield point angle diff. %f < %f
846	181	Check	Error	Step %d, Check: Torque rate %.4f > %.4f
847	182	Check	Error	Step %d, Check: Torque rate %.4f < %.4f
848	183	Check	Error	Step %d, Check: Deviation %f > %f
849	184	Check	Error	Step %d, Check: Torque Rate calculation failed
850	185	Check	Error	Step %d, Check: Overflow in Torque Rate recording buffer

Code	Comp Code	Type	Severity	Event string
851	186	Setup	Error	Step %d, Check: Torque rate/deviation has bad configuration
852	423	Setup	Info	Step %d, Retry is ordered but this step has no retry step programmed
853	426	Check	Error	Step %d, Check: Angle failed because Torque stop was never reached
854	427	Check	Error	Step %d, Monitoring: Torque level stop for Threshold angle was never reached
863	460	Setup	Error	Step %d, Control: Start cond. "Sync Pulse" requires Angle 2 as control channel
875	478	System	Error	Step %d, Received illegal Retry order in SP_RUN_STEPS from station. (Spindle running %d.)
876	479	Check	Error	Step %d, Check: Peak torque failed.
878	481	Check	Error	Step %d, Diagnostic: Static Zero Offset high on T2. abs(%f) > %f
879	482	Check	Error	Step %d, Diagnostic: Dynamic Zero Offset high on T2. abs(%f) > %f
880	483	Check	Error	Step %d, Diagnostic: Flying Zero Offset high on T2. abs(%f) > %f
881	484	Check	Warning	Step %d, Diagnostic: Flying Zero Offset high warning on T2. abs(%f) > %f
883	531	Setup	Error	Step %d, Control: Torque-Angle step has bad configuration
884	532	Setup	Error	Step %d, Restriction: Start torque for FS Angle has bad configuration
885	533	Setup	Error	Step %d, Restriction: Start condition for FS Angle has bad configuration
887	535	Check	Error	Step %d, Monitoring 2nd: Final torque could not be evaluated
888	536	Check	Error	Step %d, Monitoring 2nd: Threshold angle could not be evaluated
889	537	Check	Error	Step %d, Monitoring 2nd: Final torque. %f > %f
890	538	Check	Error	Step %d, Monitoring 2nd: Threshold angle. %f > %f
891	539	Check	Error	Step %d, Monitoring 2nd: Final torque. %f < %f
892	540	Check	Error	Step %d, Monitoring 2nd: Threshold angle. %f < %f
893	541	Check	Error	Step %d, Monitoring 2nd: Torque level stop for Threshold angle was never reached
894	542	Setup	Info	Monitoring 2nd buffer: No second channels. 2nd buffer used in check %d, step %d.
895	549	Setup	Error	Step %d, Restriction: Incorrect FS Min torque configuration
897	555	Setup	Error	Step %d, Restriction: Incorrect Gradient restriction configuration
898	683	Setup	Error	Step %d, Check: Incorrect Yield Point Check configuration

Code	Comp Code	Type	Severity	Event string
899	690	Check	Error	Step %d, Diagnostic: Spindle shunt test failed on channel %d
901	187	Check	Error	Step %d, Restriction: Fail-safe torque
902	188	Check	Error	Step %d, Restriction: Fail-safe angle
903	189	Check	Error	Step %d, Restriction: Fail-safe time
905	191	Check	Error	Step %d, Restriction: Cross thread and gradient
906	192	Check	Error	Step %d, Restriction: Torque profile
907	193	Check	Error	Double transducer: torque diff high
908	194	Check	Error	Double transducer: angle diff high
909	195	System	Error / Info	Monitoring: Overflow in recording buffer %d
910	196	Hardware	Error	Step %d, Servo hardware failure, Id %d
911	197	Setup	Error	Incorrect Servo configuration
917	203	System	Error	No Snug point was found
918	440	Check	Error	Step %d, Restriction: Remove fastener torque limit
923	529	Check	Error	Spindle protection: Max allowed torque for spindle exceeded
924	550	Check	Error	Step %d, Restriction: Min Torque
925	556	Check	Error	Step %d, Restriction: Gradient too high
926	557	Check	Error	Step %d, Restriction: Gradient too low
927	558	Check	Error	Step %d, Restriction: Gradient error as angle decreased
929	684	Check	Warning	Step %d, Check YP: Gradient calculation restarted as angle decreased
930	934	Check	Error	Step %d, Fatal continue rotation error, was not handled by reject management
931	935	Check	Error	Step %d, Restriction error during continue rotation
932	936	Setup	Warning	Step %d, Continue rotation is not allowed at step start, spindle was stopped
933	937	Setup	Warning	Step %d, Direction changed during Continue rotation
1001	553	System	Error	Trace: Unable to store new trace on TC %d, too many saved traces
1102	205	Hardware	Error	Low battery voltage on TC %d
1106	209	System	Error	System TC on addr 0x%x received setup from other system TC with addr 0x%x
1107	210	System	Error	TC %d received setup from 2 system TCs: 0x%x and 0x%x
1108	890	System	Error	TC %d Device type %d uses a serial channel, this is only allowed on PTC

Code	Comp Code	Type	Severity	Event string
1150	441	System	Error	TC %d restarted due to fatal SW error (code: %s, sub code: 0x%x)
1401	211	General	Warning	Printer on TC %d (port %d) does not support Device Command = %d
1500	212	Ext. com	Error	ID write error %d on TC %d, port %d
1501	213	Ext. com	Error	ID read error %d on TC %d, port %d
1502	214	Ext. com	Error	ID read error on TC %d, port %d. Invalid start char. ASCII = 0x%x
1503	215	Ext. com	Error	ID read error on TC %d, port %d. Unable to read tag, Data = %d
1504	216	Ext. com	Error	ID write error on TC %d, port %d. Unable to write tag, Data = %d
1506	530	Ext. com	Error	ID device on TC %d, failed to write process data
1507	886	Ext. com	Error	ID read error on TC %d, port %d. Tot. length of string was more than %d chars
1508	887	Ext. com	Error	ID read error TC %d, port %d. Start char was never received, must be ASCII=0x%x
1601	218	System	Error	Device init. error (TC %d, device %d, info %d)
1602	219	Ext. com	Error	Illegal link message received (TC %d, device %d, info %d)
1603	220	Ext. com	Error	Device is not on a PLC TC (TC %d, device %d)
1604	221	Ext. com	Error	Missing sequence number (TC %d, device %d, expected seq. no %d)
1605	222	Ext. com	Error	Received setup is to big (TC %d, device %d)
1606	891	Ext. com	Error	Device is not on the System TC (TC %d, device %d)
1701	223	System	Error	Fieldbus on TC %d has no contact with PLC
1702	224	System	Error	Fieldbus on TC %d can not read process data (err %d)
1703	225	System	Error	Fieldbus on TC %d can not write process data (err %d)
1704	226	Ext. com	Error	Fieldbus on TC %d can not read from board (err %d)
1705	227	Ext. com	Error	Fieldbus on TC %d can not write to board (err %d)
1706	228	Ext. com	Error	Fieldbus on TC %d rec. a req. (0x%x) that requires Data bytes In >= %d
1707	229	Ext. com	Error	Fieldbus on TC %d rec. a req. (0x%x) that requires Data bytes Out >= %d
1708	230	Setup	Error	Fieldbus on TC %d requires Data bytes Out > %d to auto load cycle data
1709	231	Ext. com	Error	Fieldbus on TC %d rec. a Flush/Skip req. (0x%x) for an invalid address %d
1710	232	Ext. com	Error	Fieldbus on TC %d rec. an ambig. req. (0x%x) (>1 of R, W, F & S are set)

Code	Comp Code	Type	Severity	Event string
1711	233	Ext. com	Error	Fieldbus on TC %d rec. a req. (0x%x) having an invalid extended format
1712	234	Ext. com	Error	Fieldbus on TC %d rec. a req. (0x%x) having an invalid address (%d)
1713	235	System	Error	Fieldbus on TC %d is not licensed
1901	237	System	Error	API on TC %d rec. an illegal message (err %d, info %d)
1902	238	System	Error	API on TC %d rec. a Read Setup req. Only allowed on TC 1
1903	239	System	Error	API on TC %d rec. a Write Setup req. Only allowed on TC 1
1904	240	System	Error	API on TC %d rec. a Read Setup Item req. Only allowed on TC 1
1905	241	System	Error	API on TC %d rec. a Write Setup Item req. Only allowed on TC 1
1906	242	System	Info	API on TC %d failed to send answer (cmd=%d, ret=%d)
2001	243	System	Error	CCH: Error when receiving (type=%d, d1=%d, d2=%d)
2005	247	System	Error	CCH: Unable to delete trace (first failing TC=%d)
2006	248	System	Error	CCH: Unable to delete events (first failing TC=%d)
2007	249	System	Error	CCH: Unable to delete cycle data (first failing TC=%d)
2008	250	System	Error	CCH: Unable to delete SPC data (first failing TC=%d)
2009	251	System	Error	CCH: Link message is to big (sock=%d, size=%d)
2010	252	System	Error	CCH: Link message has no header (sock=%d)
2011	892	General	Error	CCH: Function GetSetupObject() used
2100	253	System	Error	REP: Calculated size wrong (value=%f, width=%d, ndec=%d)
2101	433	Setup	Error	REP: Invalid date/time format selected
2201	254	System	Error	IO: two units with same address
2202	255	System	Error	IO: Illegal channel
2203	256	System	Error	IO: No CAN driver found
2204	257	Hardware	Error	IO: TC %d, chan %d. Subscriber not found
2205	258	Ext. com	Error	IO: Error communicating with I/O device (TC %d, identity %d)
2206	893	Ext. com	Error	IO: Error when configurating I/O signal (TC %d, MacId %d, Err %d)
2207	902	Hardware	Error	IO: Node %d, Sig %d, referring address %d on fieldbus
2400	260	System	Error	Zip expand failed. Error %d, size in %d, size out: %d
2401	261	System	Error	Zip compress failed. Error %d, size in %d, size out: %d
2701	681	Hardware	Error	No angle channel found
2702	682	Hardware	Error	No torque channel found

Code	Comp Code	Type	Severity	Event string
2802	345	System	Error	Invalid communication port %d for device
2900	347	System	Error	Unexpected signal
3100	348	System	Error	Hunt for station failed
3101	349	System	Error	TestBolt: DB read error, code %d
3102	350	Setup	Error	TestBolt: No station defined in the setup
3103	351	General	Error	TestBolt: Disabled by PLC for station %d
3201	352	System	Error	PLC: Run time system error (code 0x%x, info 0x%x)
3202	353	System	Error	PLC: Bootfile access error (error %d)
3203	354	System	Error	PLC: Only one input and one output 'EXTCOM' driver allowed (now %d and %d)
3204	355	System	Error	PLC: 'EXTCOM' area (input %d + output %d) is to big (max %d bytes)
3205	356	Setup	Error	PLC: 'ANYBUS' variables defined but no Fieldbus device exists
3206	357	Setup	Error	PLC: Only one 'ANYBUS' input/output driver is allowed (now %d and %d)
3207	358	Setup	Error	PLC: 'ANYBUS' input area (%d) is to big. Fieldbus declares only %d bytes
3208	359	General	Error	PLC: User event %d generated
3209	360	Setup	Error	PLC: 'ANYBUS' output area (%d) is to big. Fieldbus declares only %d bytes
3210	567	General	Error	PLC: The target device for DEVCMD does not exist
3211	568	General	Error	PLC: Device %d on TC %d is not allowed as target for DEVCMD (wrong type %d)
3212	569	General	Error	PLC: Failed to send DEVCMD %d to device %d on TC %d
3215	896	Setup	Error	PLC: LIO driver will not be used. Use I/O device instead.
3216	958	System	Error	PLC: Illegal channel no %d.
3303	364	Setup	Error	Fieldbus on TC %d: Config. error. Max output size=%d, max input size=%d
3305	366	Setup	Error	Fieldbus on TC %d: Board type mismatch, type connected=%d, type configured=%d
3307	368	Setup	Error	Fieldbus on TC %d: Configured Node addr.(%d) is out of range
3308	369	Setup	Error	Fieldbus on TC %d: Setting Node addr. by SW is not supported
3309	370	Setup	Error	Fieldbus on TC %d: Setting Baude Rate by SW is not supported
3310	371	Setup	Error	Fieldbus on TC %d: Configured Baude Rate (%d) is not supported
3311	372	Setup	Error	Fieldbus on TC %d: Configured Source Node addr. (%d) is out of range

Code	Comp Code	Type	Severity	Event string
3312	373	Setup	Error	Fieldbus on TC %d: Source no. of words (%d) is > Fast bytes In / 2 (%d)
3313	374	Setup	Error	Fieldbus on TC %d: Source Offset + Source no. of words (=%d) must be <= 32
3314	375	Setup	Error	Fieldbus on TC %d: Unsupported param used (Modbus specific)
3315	543	Setup	Error	Fieldbus on TC %d: Configured Number of Stations (%d) is not supported
3316	897	Setup	Error	Fieldbus on TC %d: Config error. PLC bytes in/out must be over 0.
3317	898	Ext. com .	Error	Fieldbus on TC %d: Error! Fieldbus could not be initialized!
3318	910	Setup	Error	Fieldbus on TC %d: Configuration error. (%d)!
3402	377	System	Error	ToolsNet: Illegal message received (error type %d, info %d)
3403	378	Ext.com.	Info	Toolsnet: Connection to server established using protocol ver. %f
3404	379	Ext.com.	Error	Toolsnet: Connection to server lost
3405	380	Setup	Error	ToolsNet: Bad Data IP address (%X)
3406	381	Setup	Error	ToolsNet: Bad Status IP address (%d)
3407	428	Ext. com	Info	ToolsNet: Wrong TgmId received for Ack, Exp.=%d, Rec.=%d
3408	429	Ext. com	Info	ToolsNet: Wrong SeqNo received for Ack, Exp.=%d, Rec.=%d
3409	430	Ext. com	Error	ToolsNet: Timeout when waiting for Acknowledge
3410	456	Ext. com	Error	ToolsNet: Wrong message ver., TgmId: %d, TgmVer: %d (exp.), %d (rec.)
3411	457	Ext. com	Error	ToolsNet: ToolsNet rejected protocol version %f
3412	458	Ext. com	Warning	ToolsNet: Telegram Id %d is not supported in protocol version %f
3413	459	Setup	Error	ToolsNet: Entered protocol version %f is not supported in PowerMACS
3414	622	Ext. Com.	Error	ToolsNet: Link closed by server (state %d)
3415	623	Ext. Com.	Error	ToolsNet: Link closed due to error %d (state %d)
3416	624	Ext. Com.	Error	ToolsNet: Link closed due to send congestion (error %d, state %d)
3450	601	System	Error	ToolsNet: Station number is missing in Cycle data
3451	602	System	Error	ToolsNet: Start time is missing in Cycle data
3452	603	System	Error	ToolsNet: Station status is missing in Cycle data
3453	604	System	Error	ToolsNet: Bolt number is missing in Cycle data
3454	605	System	Error	ToolsNet: Bolt status is missing in Cycle data
3455	606	System	Error	ToolsNet: Bolt data for bolt no: %d is missing in Cycle data

Code	Comp Code	Type	Severity	Event string
3456	607	System	Error	ToolsNet: Operation mode for bolt no: %d is missing in Cycle data
3457	608	System	Error	ToolsNet: Wrong format for torque and/or angle factors
3458	609	System	Error	ToolsNet: Reported data for %d of %d optional variables for Bolt: %d
3459	610	System	Error	ToolsNet: Reported data for %d of %d Bolts
3460	611	System	Error	ToolsNet: Cycle data is corrupt, not possible to send, Info = %d
3500	382	System	Error	SPC: DB read error (%d)
3501	383	System	Error	SPC: Out of memory
3503	384	System	Error	SPC: Variable does not exist (Bolt: %d, StepNo: %d, Var: %d)
3504	385	SPC	Error	SPC: One out check failed
3505	386	SPC	Error	SPC: End gather check failed
3506	387	SPC	Error	SPC: Mid gather check failed
3507	388	SPC	Error	SPC: Seven above check failed
3508	389	SPC	Error	SPC: Seven below check failed
3509	390	SPC	Error	SPC: Seven up check failed
3510	391	SPC	Error	SPC: Seven down check failed
3511	392	SPC	Error	SPC: Cp limit check failed
3512	393	SPC	Error	SPC: Cpk limit check failed
3513	394	SPC	Error	SPC: Cam limit check failed
3514	395	System	Error	SPC: Sub group size must be > 0 (NoSttSgSave=%d Sgsize=%d NoSubSgSave=%d)
3515	396	System	Error	SPC: Ref. to free area is corrupt (FreeMem=%d SubgrpBase=%d NoSubSgSave=%d)
3516	397	SPC	Error	SPC: SPC data structure corrupt, pos %d, idx: %d, %d
3517	398	Setup	Error	SPC: Too many variables for spindle %d
3601	399	Ext. com	Error	Illegal link message rec. (device %d, type %d, info %d)
3700	400	Ext. com	Error	Send failed. Not possible to correct with retransm. (port %d, info %d)
3701	401	Ext. com	Error	Received failed (port %d, info %d)
3702	402	Ext. com	Error	State machine error for send (port %d, state %d)
3703	403	Ext. com	Error	State machine error for receive (port %d, state %d)
3704	404	Ext. com	Error	Status from sio not OK (port %d, retcode %d)
3705	405	Ext. com	Error	Unknown protocol selected (port %d, protocol type %d)
3706	406	Ext. com	Error	Bad Checksum received (port %d, Checksum in tgm %d, calc. Checkum %d)

Code	Comp Code	Type	Severity	Event string
3707	407	Ext. com	Error	Read telegram failed (TELWAY protocol, info %d)
3801	408	System	Error	EPH: Error when receiving (type=%d, d1=%d, d2=%d)
3802	409	System	Error	EPH: Error when sending (type=%d, d1=%d, d2=%d)
3811	420	Ext. com	Info	EPH: Connection established
3812	421	Ext. com	Error	EPH: Connection lost
3813	422	Setup	Error	EPH: Bad Data IP address, %d
3814	505	Ext. com	Error	EPH: Bad configuration of Barcode and VIN number
3815	506	Ext. com	Error	EPH: Station QO has not correct value: %d
3816	546	Ext. com	Error	EPH: Bad length of received TMU, (acctual = %d, expected = %d)
3817	508	Ext. com	Error	EPH: No response from PLUS server
3818	509	Ext. com	Error	EPH: Wp.Id has bad layout (Length = %d)
3819	510	Ext. com.	Info	EPH: Connection to PLUS established as server
3820	511	Ext. com.	Info	EPH: Connection to PLUS established as client
3821	519	Ext. com.	Error	EPH: No TMU received from PLUS, NNEG received
3822	520	Ext. com.	Info	EPH: No TMU received from PLUS, No tightening shall be made for sent barcode
3823	521	Ext. com.	Warning	EPH: No TMU received from PLUS, Barcode already used
3824	522	Ext. com.	Error	EPH: No TMU received from PLUS, telegram sent from TC inconsistent
3825	523	Ext. com.	Warning	EPH: Problem in PLUS server, NNEG received in state Idle
3826	524	Ext. com.	Warning	EPH: No matching mode number for received TMU
3827	525	Ext. com.	Error	EPH: No SA found for mode number: %d
3828	526	Ext. com.	Error	EPH: SA has not correct value: %d
3829	529	Ext. com.	Warning	EPH: No TMU received from PLUS, Double tightening not allowed
3830	530	Ext. com.	Error	EPH: No TMU received from PLUS, Id No unknown in PLUS server
3831	547	Ext. com.	Error	EPH: No TMU received from PLUS, Bad layout of telegram
3832	548	Ext. com.	Error	EPH: No TMU received, no connection to PLUS
3833	507	Ext. com.	Error	EPH: Request of TMU failed, barcode has bad length (actual = %d, expected = %d)
3834	570	Ext. com.	Error	EPH: Data could not be sent, NACK received from I-P.M.
3835	571	Ext. com.	Error	EPH: Data could not be sent, no answer from I-P.M.
3836	572	System	Info	EPH: Received TMU from PLUS not accepted, device already busy

Code	Comp Code	Type	Severity	Event string
3837	573	System	Info	EPH: Received TMU from PLUS not accepted, device busy sending cycledata
3838	574	System	Error	EPH: A non VBA station received a TMU without a request from the TC
3839	589	Ext. com.	Warning	EPH: No program number defined, used PgmNo = 0 when sending data to I-P.M.
3840	590	Ext. com.	Warning	EPH: The AFO no was too long, %d chars, only %d sent to I-P.M.
3841	614	Ext. com	Error	EPH: No TMU received from PLUS, wrong IT type in received telegram
3843	616	Ext. com	Info	EPH: Total length of IT and IW is %d chars. Truncated after %d chars
3845	618	Ext. com	Error	EPH: Send data failed, Wp ID must start with specified VIN number type
3846	619	Ext. com	Error	EPH: Send data failed, not possible to store data in PLUS (TC received I*NQ)
3847	620	Ext. com	Error	EPH: Unknown or corrupt tgm received from PLUS (Error=%d, tgm id=%d, state=%d)
3848	621	Ext. com	Error	EPH: Error when sending telegram to PLUS (Error=%d, tgm id=%d, state=%d)
3850	719	System	Error	EPH: No connection to station %d
3851	720	System	Error	EPH: Group %d cannot be selected/deselected since it does not exist
3852	721	System	Error	EPH: Chan. %d cannot be selected since it does not exist
3853	722	System	Error	EPH: Group %d cannot be deselected since it is not selected
3854	723	System	Error	EPH: Group %d is not valid for the operation
3855	724	System	Error	EPH: TAP/TNR cannot be added to chan. %d since it is not part of group %d
3856	725	System	Error	EPH: Cannot enable group %d since it is not selected (no GRS.GRP received)
3857	726	System	Error	EPH: Cannot enable chan. %d since it is not selected (no PRS received)
3858	727	General	Info	EPH: Start failed since no groups or channels are specified
3859	728	Ext. Com.	Error	EPH: Too large XML message
3860	729	System	Warning	EPH: Master PC PNR %d not supported. PNR of Audi XML device is %d
3861	730	System	Error	EPH: Cannot disable group %d since it is not enabled
3862	731	System	Error	EPH: Cannot disable chan. %d since it is not enabled

Code	Comp Code	Type	Severity	Event string
3863	732	System	Error	EPH: Cannot disable chan. %d of group %d since it is not part of the group
3864	733	System	Error	EPH: Group %d cannot be started since it is not enabled
3865	734	System	Error	EPH: Channel %d cannot be started since it is not enabled
3866	735	System	Error	EPH: Start failed since group %d is already running
3867	736	System	Error	EPH: Start failed since chan. %d is already running
3868	737	System	Error	EPH: Start failed since station %d used by group %d already is occupied
3869	738	System	Error	EPH: Start failed since station %d used by chan. %d already is occupied
3871	739	System	Error	EPH: Start failed since groups does not have the same batch parameters
3872	740	System	Error	EPH: Start failed since chan. %d of group %d already is occupied
3873	741	System	Error	EPH: Start failed since chan. %d already is occupied
3874	742	System	Error	EPH: Start/stop of station %d failed due to connection problem
3875	743	System	Error	EPH: Group %d cannot be selected since it does not have ny channels
3876	744	System	Info	EPH: Manual start ignored since device is not configured for manual control
3877	745	System	Info	EPH: Manual loosen ignored since it is not enabled by the Master PC
3878	746	System	Error	EPH: Chan. %d cannot be selected since there is no program with no. %d
3879	747	System	Error	EPH: %s without KNR is not supported
3880	748	System	Error	EPH: %s without GRS and KNR is not supported
3881	749	System	Error	EPH: %s with both GRS and KNR is not supported
3882	750	System	Error	EPH: Start of station %d failed. Machine stopping all stations
3883	911	System	Error	EPH: VW Variant not allowed on systems with more then one TC
3901	416	Setup	Error	GM DeviceNet: Input area (%d) is wrong size. Fieldbus declares only (%d) bytes
3902	417	Setup	Error	GM DeviceNet: Output area (%d) is wrong size. Fieldbus declares only (%d) bytes
3903	418	System	Error	Hunt for station failed
4013	685	Check	Error	Step %d, Check: Yield Point check failed, no yield point found
4015	686	Check	Error	Step %d, Check: Yield Point check failed, not possible to measure angle

Code	Comp Code	Type	Severity	Event string
4016	687	Check	Error	Step %d, Check: Yield Point Angle high, %f > %f
4017	688	Setup	Error	Step %d, Check: Yield Point Angle low, %f < %f
4018	689	Check	Error	Step %d, Check: Stick-Slip has bad configuration
4101	476	Ext. com.	Info	ACTA: Batch result received. (torque flags=%d, angle flags=%d, pulses flags=%d)
4103	595	Ext. com.	Warning	ACTA: Set Torq SF failed for sp %d, too high diff (old = %f, new = %f)
4104	596	Ext. com.	Error	ACTA: Set Torq Scale Factor failed for sp %d
4105	597	Ext. com.	Info	ACTA: Torq SF changed for T1 on spindle %d, old = %f, new = %f
4106	598	Ext. com.	Info	ACTA: DATAHOLD or DATADROP used, no cycle data will be avaliable for ACTA
4107	599	Ext. com.	Warning	ACTA: Set Torq SF failed for spindle %d, it is used more than once in the cycle
4108	600	Ext. com.	Info	ACTA: Not possible to start tightening, spindle %d is not used in mode %d
4201	650	Ext. com.	Error	DASP: TC %d: Message receiving error (0x%x)
4203	651	Ext. com.	Error	DASP: TC %d: Received wrong DLP task (task=%d), (key=%d)
4206	654	Ext. com.	Error	DASP: Servo on TC:%d has wrong LLP version (0x%x)
4207	655	Ext. com.	Error	DASP: Spindle on TC:%d has wrong LLP version (0x%x)
4208	656	Ext. com.	Error	DASP: TC %d: No servo connected
4209	657	Ext. com.	Error	DASP: TC %d: No spindle connected
4210	658	Ext. com.	Error	DASP: TC %d: Stream %d resend max time
4211	659	Ext. com.	Info	DASP: TC %d: New software has been downloaded to the servo
4212	660	Ext. com.	Info	DASP: TC %d: New software has been downloaded to the spindle
4213	661	System	Error	DASP: TC %d: Update EEPROM in servo, command has already been sent
4214	662	System	Error	DASP: TC %d: Bank select failed
4215	663	System	Error	DASP: TC %d: Failed to read from device boot
4216	664	System	Error	DASP: TC %d: Application do not match this device (Model %d, App %d)
4217	665	System	Error	DASP: TC %d: Device HW version too low (Variant %d, LowLim %d)
4218	666	System	Error	DASP: TC %d: Device HW version too high (Variant %d, HighLim %d)
4219	667	Ext. com.	Error	DASP: TC %d: Servo ALP version too high (Servo %d, Supported up to %d)

Code	Comp Code	Type	Severity	Event string
4220	668	Ext. com.	Error	DASP: TC %d: Spindle ALP version too high (Spindle %d, Supported up to %d)
4223	671	Ext. com.	Error	DASP: TC %d: Update EEPROM in spindle, command has already been sent
4224	648	Ext. com.	Info	DASP: TC %d: Restarting servo and TC.
4225	649	Ext. com.	Info	DASP: TC %d: Restarting spindle.
4227	677	General	Error	TC %d: Incorrect spindle article number
4228	678	General	Warning	TC %d: Any Spindle article number in use
4229	679	General	Error	TC %d: Simulated Spindle and Servo in use
4230	680	System	Info	TC %d, Replaced ramp down with i. brake (too short time between start/stop)
4231	903	General	Info	Conv. Box: TC %d, New box detected
4232	904	General	Error	Conv. Box: TC %d, Failed to program box with reference values from setup
4233	905	General	Error	Conv. Box: TC %d, Motortune is required and calibration is recommended
4234	906	General	Info	Conv. Box: TC %d, Box programmed successfully
4235	907	General	Warning	Conv. Box: TC %d, Checksum faults detected. Verify spindle calibration values
4401	554	General	Warning	Multiple ID: Bypassed Work Order position %d, which uses Identifier Type %d
4501	703	Ext. com.	Error	ANYBUS: TC %d: Unable to initialize the serial channel
4502	704	Ext. com.	Error	ANYBUS: TC %d: Unable to configure the serial channel
4503	705	Ext. com.	Error	ANYBUS: TC %d: Driver Error! Severity (%d) Code (%d)
4504	706	Ext. com.	Warning	ANYBUS: TC %d: Connection to ABCC Recovered.
4505	707	Ext. com.	Info	ANYBUS: TC %d: Connection to ABCC lost!
4507	708	Ext. com.	Error	ANYBUS: TC %d: Error! No module connected!
4508	709	Ext. com.	Error	ANYBUS: TC %d: Error! Module not supported!
4509	710	Ext. com.	Error	ANYBUS: TC %d: Error! Wrong anybus network module mounted!
4510	711	Ext. com.	Error	ANYBUS: TC %d: Error! Attached module is not an AnyBusCC module!
4511	712	Ext. com.	Error	ANYBUS: TC %d: Error! Exception (%d) in AnyBus module!
4512	713	Ext. com.	Error	ANYBUS: TC %d: Error! Read request offset (%d) is out of range!
4513	714	Ext. com.	Error	ANYBUS: TC %d: Error! Write request offset (%d) is out of range!

Code	Comp Code	Type	Severity	Event string
4514	715	Ext. com.	Error	ANYBUS: TC %d: Error! Failed to set up read map!
4515	716	Ext. com.	Error	ANYBUS: TC %d: Error! Failed to set up write map!
4516	717	Ext. com.	Error	ANYBUS: TC %d: Setup done!
4517	718	Ext. com.	Error	ANYBUS: TC %d: Error! Module version (%x) does not support Extended Mode!
5001	751	Hardware	Error	TC %d, Servo Overtemp: Power Stage %f deg
5002	752	Hardware	Error	TC %d, Servo EEPROM Version Error
5003	753	Hardware	Error	TC %d, Servo Motor parameters invalid
5004	754	Hardware	Error	TC %d, Servo Motortune parameters invalid
5005	755	Hardware	Error	TC %d, Servo software error Id %d Code %d
5006	756	Hardware	Error	TC %d, Servo software warning Id %d Code %d
5007	757	Hardware	Error	TC %d, Servo Ref21 Error 0x%x
5020	758	Hardware	Error	TC %d, Spindle Overtemp: Motor %f deg, Chip %f deg
5021	759	Hardware	Error	TC %d, Spindle 5V Digital Torque Off
5022	760	Hardware	Error	TC %d, T1 Voltage Off
5023	761	Hardware	Error	TC %d, Spindle EEPROM Error. New spindle parameters needed.
5024	762	Hardware	Error	TC %d, Torque 1 Low
5025	763	Hardware	Error	TC %d, Torque 1 High
5026	764	Hardware	Error	TC %d, Torque 2 Low
5027	765	Hardware	Error	TC %d, Torque 2 High
5028	766	Hardware	Error	TC %d, Torque 2 Voltage Off
5029	767	Hardware	Error	TC %d, Spindle Chassis ground error
5030	768	Hardware	Error	TC %d, Spindle Ref21 Error 0x%x
5040	769	System	Error	TC %d, Could not read ref %d from spindle
5041	770	System	Error	TC %d, Could not read ref %d from servo
5042	771	Hardware	Error	Step %d, Diagnostic: Shunt values T1 Offset %d, Gain %d
5043	772	Hardware	Error	Step %d, Diagnostic: Shunt values T2 Offset %d, Gain %d
5044	773	System	Error	TC %d, Could not write ref %d to spindle, bytes %d
5045	774	System	Error	TC %d, Could not write ref %d to servo, bytes %d
5046	775	System	Error	TC %d, Failed to update checksum group %d on spindle
5047	776	System	Error	TC %d, Failed to update checksum group %d on servo
5048	777	System	Error	TC %d, Channel %d Hardware Error %x
5049	778	General	Info	TC %d, Motortune Started

Code	Comp Code	Type	Severity	Event string
5050	779	General	Info	TC %d, Motortune Complete
5051	780	General	Info	TC %d, Motortune Aborted
5052	781	General	Info	TC %d, Motortune Failed, Progress %x, Status %x
5053	782	Hardware	Error	TC %d, Drive parameters invalid
5054	783	System	Error	TC %d, Channel error %x
5055	784	Check	Error	Diagnostic: Torque/Current calibration finished due to time
5056	785	System	Error	Servo Communication Error (WD %d)
5070	786	Hardware	Error	TC %d, Spindle Software Warning
5071	787	Hardware	Error	TC %d, Spindle Software Error
5080	788	Hardware	Error	TC %d, Servo Current Control Error
5081	789	Hardware	Error	TC %d, Servo Driver Stalled
5082	790	Hardware	Error	TC %d, Servo Missed angle
5083	791	Power	Error	TC %d, Gate Driver Voltage Low
5090	792	Hardware	Error	TC %d, Servo Current High
5091	793	Hardware	Error	TC %d, Servo Offset Error
5092	794	Hardware	Error	TC %d, Servo HW Trip
5093	795	Hardware	Error	TC %d, Servo I2R Integrator High
5094	796	Hardware	Error	TC %d, Servo DC Bus Low
5095	797	Hardware	Error	TC %d, Servo DC Bus High
5096	798	Setup	Warning	Negative spindle windup.
5101	799	System	Error	TC %d, Spindle Functional Test Required (Diagnostic Step)
5102	800	System	Info	TC %d, FPGA ERROR Isr & Imr 0x%x Usr 0x%x
5103	801	System	Info	TC %d, FPGA OVERFLOW Count %d Usr 0x%x
5104	802	Hardware	Error	TC %d, Sensor Error on Angle Channel %d
5105	803	Hardware	Error	TC %d, Sensor Error on Torque Channel %d
5201	804	System	Error	TC %d, Error writing serial number to spindle. Ref %d, Status %d
5202	805	System	Error	TC %d, Error reading serial number from spindle. Ref %d, Status %d
5203	806	System	Error	TC %d, Error writing model to spindle. Ref %d, Status %d
5204	807	System	Error	TC %d, Error reading model from spindle. Ref %d, Status %d
5205	808	System	Error	TC %d, Error writing type to spindle. Ref %d, Status %d
5206	809	System	Error	TC %d, Error reading type from spindle. Ref %d, Status %d
5207	810	System	Error	TC %d, Error writing max torque to spindle. Ref %d, Status %d

Code	Comp Code	Type	Severity	Event string
5208	811	System	Error	TC %d, Error reading max torque from spindle. Ref %d, Status %d
5209	812	System	Error	TC %d, Error writing gear ratio to spindle. Ref %d, Status %d
5210	813	System	Error	TC %d, Error reading gear ratio from spindle. Ref %d, Status %d
5211	814	System	Error	TC %d, Error writing motor temp to spindle. Ref %d, Status %d
5212	815	System	Error	TC %d, Error reading motor temp from spindle. Ref %d, Status %d
5213	816	System	Error	TC %d, Error writing electronics temp to spindle. Ref %d, Status %d
5214	817	System	Error	TC %d, Error reading electronics temp from spindle. Ref %d, Status %d
5215	818	System	Error	TC %d, Error writing units per lap to spindle. Ref %d, Status %d
5216	819	System	Error	TC %d, Error reading units per lap from spindle. Ref %d, Status %d
5217	820	System	Error	TC %d, Error writing torque transducer config to spindle. Ref %d, Status %d
5218	821	System	Error	TC %d, Error writing angle transducer config to spindle. Ref %d, Status %d
5219	822	System	Error	TC %d, Error reading torque transducer config from spindle. Ref %d, Status %d
5220	823	System	Error	TC %d, Error reading angle transducer config from spindle. Ref %d, Status %d
5221	824	System	Error	TC %d, Error writing angle diff config to spindle. Ref %d, Status %d
5222	825	System	Error	TC %d, Error reading angle diff config from spindle. Ref %d, Status %d
5223	826	System	Error	TC %d, Error writing torque diff config to spindle. Ref %d, Status %d
5224	827	System	Error	TC %d, Error reading torque diff config from spindle. Ref %d, Status %d
5225	828	System	Error	TC %d, Error writing t/c factor to spindle. Ref %d, Status %d
5226	829	System	Error	TC %d, Error reading t/c factor from spindle. Ref %d, Status %d
5227	830	System	Error	TC %d, Error writing torque amplification to spindle. Ref %d, Status %d
5228	831	System	Error	TC %d, Error reading torque amplification from spindle. Ref %d, Status %d
5229	832	System	Error	TC %d, Error writing calibration value to spindle. Ref %d, Status %d

Code	Comp Code	Type	Severity	Event string
5230	833	System	Error	TC %d, Error reading calibration value from spindle. Ref %d, Status %d
5231	834	System	Error	TC %d, Error writing calibration date to spindle. Ref %d, Status %d
5232	835	System	Error	TC %d, Error reading calibration date from spindle. Ref %d, Status %d
5233	836	System	Error	TC %d, Error writing motortune params to spindle. Ref %d, Status %d
5234	837	System	Error	TC %d, Error reading motortune params from spindle. Ref %d, Status %d
5235	838	System	Error	TC %d, Error writing motor params to spindle. Ref %d, Status %d
5236	839	System	Error	TC %d, Error reading motor params from spindle. Ref %d, Status %d
5237	840	System	Error	TC %d, Error writing commutation offset to spindle. Ref %d, Status %d
5238	841	System	Error	TC %d, Error reading commutation offset from spindle. Ref %d, Status %d
5239	842	System	Error	TC %d, Error writing date and time to spindle. Ref %d, Status %d
5240	843	System	Error	TC %d, Error reading date and time from spindle. Ref %d, Status %d
5241	844	System	Error	TC %d, Error writing service date to spindle. Ref %d, Status %d
5242	845	System	Error	TC %d, Error reading service date from spindle. Ref %d, Status %d
5243	846	System	Error	TC %d, Error writing service interval to spindle. Ref %d, Status %d
5244	847	System	Error	TC %d, Error reading service interval from spindle. Ref %d, Status %d
5245	848	System	Error	TC %d, Error writing cycles since service to spindle. Ref %d, Status %d
5246	849	System	Error	TC %d, Error reading cycles since service from spindle. Ref %d, Status %d
5247	850	System	Error	TC %d, Error writing cycle counter to spindle. Ref %d, Status %d
5248	851	System	Error	TC %d, Error reading cycle counter from spindle. Ref %d, Status %d
5249	852	System	Error	TC %d, Error writing last shunt calibration to spindle. Ref %d, Status %d

Code	Comp Code	Type	Severity	Event string
5250	853	System	Error	TC %d, Error reading last shunt calibration from spindle. Ref %d, Status %d
5251	854	System	Error	TC %d, Error writing wind up to spindle. Ref %d, Status %d
5252	855	System	Error	TC %d, Error reading wind up from spindle. Ref %d, Status %d
5253	856	System	Error	TC %d, Error writing hardware revision to spindle. Ref %d, Status %d
5254	857	System	Error	TC %d, Error reading hardware revision from spindle. Ref %d, Status %d
5255	858	System	Error	TC %d, Error writing bootloader to spindle. Ref %d, Status %d
5256	859	System	Error	TC %d, Error reading bootloader from spindle. Ref %d, Status %d
5257	860	System	Error	TC %d, Error writing application to spindle. Ref %d, Status %d
5258	861	System	Error	TC %d, Error reading application from spindle. Ref %d, Status %d
5259	862	System	Error	TC %d, Error writing chip serial no to spindle. Ref %d, Status %d
5260	863	System	Error	TC %d, Error reading chip serial no from spindle. Ref %d, Status %d
5261	908	System	Error	TC %d, Error writing reference to spindle. Ref %d, Status %d
5262	909	System	Error	TC %d, Error reading reference from spindle. Ref %d, Status %d
5301	864	System	Error	TC %d, Error writing power stage temperature to servo. Ref %d, Status %d
5302	865	System	Error	TC %d, Error reading power stage temperature from servo. Ref %d, Status %d
5303	866	System	Error	TC %d, Error writing torque constant to servo. Ref %d, Status %d
5304	867	System	Error	TC %d, Error reading torque constant from servo. Ref %d, Status %d
5305	868	System	Error	TC %d, Error writing conversion constants to servo. Ref %d, Status %d
5306	869	System	Error	TC %d, Error reading conversion constants from servo. Ref %d, Status %d
5307	870	System	Error	TC %d, Error writing motortune parameters to servo. Ref %d, Status %d
5308	871	System	Error	TC %d, Error reading motortune parameters from servo. Ref %d, Status %d
5309	872	System	Error	TC %d, Error writing motortune status to servo. Ref %d, Status %d

Code	Comp Code	Type	Severity	Event string
5310	873	System	Error	TC %d, Error reading motortune status from servo. Ref %d, Status %d
5311	874	System	Error	TC %d, Error writing motor parameters to servo. Ref %d, Status %d
5312	875	System	Error	TC %d, Error reading motor parameters from servo. Ref %d, Status %d
5313	876	System	Error	TC %d, Error writing date and time to servo. Ref %d, Status %d
5314	877	System	Error	TC %d, Error reading date and time from servo. Ref %d, Status %d
5315	878	System	Error	TC %d, Error writing hardware revision to servo. Ref %d, Status %d
5316	879	System	Error	TC %d, Error reading hardware revision from servo. Ref %d, Status %d
5317	880	System	Error	TC %d, Error writing bootloader to servo. Ref %d, Status %d
5318	881	System	Error	TC %d, Error reading bootloader from servo. Ref %d, Status %d
5319	882	System	Error	TC %d, Error writing application to servo. Ref %d, Status %d
5320	883	System	Error	TC %d, Error reading application from servo. Ref %d, Status %d
5321	884	System	Error	TC %d, Error writing serial number to servo. Ref %d, Status %d
5322	885	System	Error	TC %d, Error reading serial number from servo. Ref %d, Status %d
5700	691	Setup	Error	Step %d, Angle channel for check (%d) has bad configuration (%d)
5701	692	Setup	Error	Step %d, Channel Angle%d for check (%d) does not exists in spindle
5702	693	Setup	Error	Step %d, Torque channel for check (%d) has bad configuration (%d)
5703	694	Setup	Error	Step %d, Channel Torque%d for check (%d) does not exists in spindle
5704	695	Setup	Error	Step %d, Angle channel for restriction (%d) has bad configuration (%d)
5705	696	Setup	Error	Step %d, Channel Angle%d for restriction (%d) does not exists in spindle
5706	697	Setup	Error	Step %d, Torque channel for restriction (%d) has bad configuration (%d)
5707	698	Setup	Error	Step %d, Channel Torque%d for restriction (%d) does not exists in spindle
5708	699	Setup	Error	Step %d, Angle channel for steptype (%d) has bad configuration (%d)

Code	Comp Code	Type	Severity	Event string
5709	700	Setup	Error	Step %d, Channel Angle%d for steptype (%d) does not exists in spindle
5710	701	Setup	Error	Step %d, Torque channel for steptype (%d) has bad configuration (%d)
5711	702	Setup	Error	Step %d, Channel Torque%d for steptype (%d) does not exists in spindle
5712	592	Setup	Error	Cust. error series bad config. Step %d, Series %d, BoltStatus %d. Series 1 used
5713	612	Setup	Error	Cust. error series bad config. Step %d, Series %d, BoltStatus %d. Series 1 used
5714	613	Setup	Error	No step failed, used customer step name in first CE step (step %d) instead
5715	888	Setup	Error	Step %d, Control: Local DI %d cannot be used since TC type is not 'PTC'
5716	912	Check	Error	Step %d, BCT measutement failed, old values are cleared
5717	913	Setup	Info	Backlash Angle Chan %d differ from stored chan %d, backlash values are cleared
5718	914	Setup	Error	Step %d, Control: Speed or speed ramp has bad configuration
5719	915	Setup	Error	Step %d, Control: The selected position target has no stored value
5720	916	Setup	Error	Step %d, Control: No prev. step direction stored, not possibel to run POS step
5721	918	Setup	Error	Step %d, Zone has bad configuration
5722	919	Check	Error	Step %d, Check: Peak torque failed, Zone %d was never started
5723	920	Check	Error	Step %d, Check: Torque in angle window failed, Zone %d was never started
5724	921	Check	Error	Step %d, Check: Torque in time window failed, Zone %d was never started
5725	922	Check	Error	Step %d, Check: Mean torque failed, Zone %d was was not finished
5726	923	Check	Error	Step %d, Check: Angle failed, Zone %d was was not finished
5727	924	Check	Error	Step %d, Check: Time failed, Zone %d was never started
5728	925	Check	Error	Step %d, Check: T/T3 failed, Zone %d was was not finished
5729	926	Check	Error	Step %d, Check: Torque Rate calculation failed, Zone %d was never started
5730	927	Setup	Error	Step %d, Res var have bad configuration. %d is not a valid cycle data variable
5731	928	Setup	Error	Step %d, Check with identity %d has bad configuration
5732	929	Check	Error	Step %d, Check: Delta torque failed, Zone %d was never started

Code	Comp Code	Type	Severity	Event string
5733	930	Check	Error	Step %d, Check: Delta torque failed
5734	931	Check	Error	Step %d, Check: Delta torque. %f > %f
5735	932	Check	Error	Step %d, Check: Delta torque. %f < %f
5736	933	Setup	Error	Step %d has bad configuration, one or more paramters are faulty, Step Type = %d
5737	938	Setup	Warning	Step %d, The step is syncronized since Wait for PLC to continue is checked.
5738	939	Pos. reg	Info	Step %d, PosValid %d, Stored External Postion = %f
5739	940	Abs. pos	Info	Acctual Internal position = %d, Acctual External position = %f
5740	941	Setup	Error	Step %d, Store of position register not possible, Angle channel does not exist
5741	942	Emergency	Error	The e-stop chain is broken somewhere between TC %d and the primary TC (TC %d)
5742	943	Emergency	Error	E-stop is faulty. TC %d (station TC) is e-stopped, some other TCs isn't
5743	944	Check	Error	Step %d, Check: Shut Off Torque failed, Zone %d was was not finished
5744	945	Check	Error	Step %d, T1 Shunt Cal NOK. Shunt drift from last cal too high, %f > %f
5745	946	Check	Error	Step %d, T2 Shunt Cal NOK. Shunt drift from last cal too high, %f > %f
5746	947	Check	Error	Step %d, T1 Zero Offset comp NOK. Drift from last cal too high %f > %f
5747	948	Check	Error	Step %d, T2 Zero Offset comp NOK. Drift from last cal too high %f > %f
5748	949	Check	Error	Step %d, T1 Shunt Calibration NOK. Measured Shunt too low %f < %f
5749	950	Check	Error	Step %d, T2 Shunt Calibration NOK. Measured Shunt too low %f < %f
5750	951	Check	Error	Step %d, T1 Shunt Calibration NOK. Measured Shunt too high %f > %f
5751	952	Check	Error	Step %d, T2 Shunt Calibration NOK. Measured Shunt too high %f > %f
5752	953	Check	Error	Step %d, T1 Zero Offset compensation NOK. Measured offset too low %f < %f
5753	954	Check	Error	Step %d, T2 Zero Offset compensation NOK. Measured offset too low %f < %f
5754	955	Check	Error	Step %d, T1 Zero Offset compensation NOK. Measured offset too high %f > %f

Code	Comp Code	Type	Severity	Event string
5755	956	Check	Error	Step %d, T2 Zero Offset compensation NOK. Measured offset too high %f > %f
5756	957	Setup	Error	Step %d, RampsAndOther: Digital signals for step start or step end has bad configuration.
5757	917	Check	Error	Step %d, Check Position: The selected position target has no stored value
5758	894	Check	Error	Step %d, Check: Distance to abs position high. %f > %f
5759	895	Check	Error	Step %d, Check: Distance to abs position low. %f < %f
5760	410	Check	Error	Step %d, Check: Low spot torque failed, Zone %d was never started
5761	411	Check	Error	Step %d, Check: Low spot torque failed, start angle was never reached
5762	412	Check	Error	Step %d, Check: Low Spot Torque High %f > %f
5763	413	Check	Error	Step %d, Check: Low Spot Torque Low %f < %f
8001	899	General	Info	TFTP: Software update of TC %d, failed

10.2 ToolsTalk PowerMACS command line

ToolsTalk PowerMACS supports the following command line arguments:

```
PowerMACS_4000.exe [/t=<ip_address>]
                   [/g=online | /g=offline]
                   [/o=backoff | /o=override]
                   [/c=target | /c=file]
                   [<setup>]
```

The arguments have the following meaning:

Argument	Usage
<setup>	Specifies the setup to open. <setup> may include a path. The directory of the PowerMACS application is used as default directory. If the file pointed out by <setup> cannot be located or read ToolsTalk PowerMACS will stop and display an error message.
/t=<ip_address>	Sets <ip_address> as the IP-address of the selected target system for ToolsTalk PowerMACS in the system register. If not specified the IP-address of the system most recently connected to is used.
/g=online /g=offline	Specifies if ToolsTalk PowerMACS should connect or not against the target system known by ToolsTalk PowerMACS. If not specified the same on-line state as ToolsTalk PowerMACS had when it was last closed is assumed. If ToolsTalk PowerMACS should go on-line but fails to connect to the target it will stop and display an error message.
/o=backoff /o=override	Specifies how to handle the situation that another ToolsTalk PowerMACS already is connected to the target system when going on-line. Only used if ToolsTalk PowerMACS is configured to go on-line.
/c=target /c=file	Specifies how to handle conflicting setups, that is, the case when the setup read from file is different compared to the one in the target system. It is only used if ToolsTalk PowerMACS is configured to go on-line. Choose "target" to ToolsTalk PowerMACS fetch the setup from the target system. Choose "file" to ToolsTalk PowerMACS to overwrite the setup read from file to the target system. If not specified "target" is assumed.

10.3 Cycle Data Storage

This scenario exemplifies the number of cycle datas possible to store given a typical reporter configuration. The setup includes a ToolsNet device and an Ethernet device of type DC PLUS.

Station Data	**Bolt Data**		**Step Data**
Station number	Op Mode	Failing Step No	Step No
Cycle Start Time	Bolt No	Bolt Pgm Strategy	Peak T
Status	Status	Spindle No	A
Data no station	Program Name	Bolt T	Shut Off T
Wp. Id	Ordinal No	Bolt A	Peak T3
Mode No	Errors	Bolt Name	
Station Name	RM Errors	Program number	
Mode Name	Warnings	Limits (Bolt T)	
Station QO	Compact Errors	Limits (Bolt A)	
Station SA		Bolt Name	
Station AB		Cust Error	
Id Res 4			

Amount of SRAM used for Cycle Data	**System with 2 bolts**	**System with 8 bolts**	**System with 16 bolts**
30 % (Default)	900 cycles	280 cycles	140 cycles
75 %	2400 cycles	700 cycles	360 cycles

10.3.1 Store 5000 Cycle Data

It is possible to store more than 5000 cycle data by distribute the storage to all TCs included in a system. The cycle data for each bolt is stored locally on the TC the bolt was run on. To be able to store more than 5000 cycle data the size of the cycle data storage has to be increased on the expense of trace data storage (there will still be room for more than 20+ trace).

Please see chapter SRAMfor more information.

10.3.1.1 Storing and retrieving cycle data

Since the cycle data for each bolt is stored locally on the TC it was run on the amount of cycle data that could be stored depends only on the size of the cycle data storage and on how many variables that is included in the configured reporters.

To minimize the need for fetching bolt cycle data from each TC when a device is requesting it a complete cycle data is stored in a volatile memory cache on TC1. This cache is filled with cycle data when a cycle is run or when a requested cycle data has been retrieved from each TC. After a power up the cache is empty an all requested cycle data has to be retrieved from each bolt.

It is possible to change TCs within a system and still be able to retrieve cycle data from them.

If however any TC should be missing or disconnected when a cycle data is fetched the data for the bolt that belong to that TC is reconstructed using status information that is saved in the cycle data header for each bolt in the cycle data. Since this data is saved together with station data and data for bolt 1 on TC1 it is kept to a minimum, the data saved for each bolt is:

- Ordinal bolt number
- Status (OK,NOK etc.)
- Failing step number (if status NOK)

A new bolt variable “**Data missing**” has been added to make it possible to detect if data was missing for a particular bolt. This variable is 1 if data is missing and 0 if not.

Appendix

Example of cycle data with all bolt data and with bolt data 5 missing:

Data No Station: 1 Station: Stn 01 2008-03-04 09:34:07 Status: OK No Bolts: 7

Ordinal No	Bolt	Data Missing	Status	Bolt T	Bolt A	Failing Step No
1	Bolt 01	0	OK	0,75	362,37	
2	Bolt 02	0	OK	0,75	362,37	
3	Bolt 03	0	OK	0,75	362,37	
4	Bolt 04	0	OK	0,75	362,37	
5	Bolt 05	0	OK	0,75	362,37	
6	Bolt 06	0	OK	0,75	362,37	
7	Bolt 07	0	OK	0,75	362,37	

Data No Station: 2 Station: Stn 01 2008-03-04 09:34:08 Status: OK No Bolts: 7

Ordinal No	Bolt	Data Missing	Status	Bolt T	Bolt A	Failing Step No
1	Bolt 01	0	OK	0,75	362,37	
2	Bolt 02	0	OK	0,75	362,37	
3	Bolt 03	0	OK	0,75	362,37	
4	Bolt 04	0	OK	0,75	362,37	
5		1	OK			
6	Bolt 06	0	OK	0,75	362,37	
7	Bolt 07	0	OK	0,75	362,37	

10.4 Configuration of the Conversion Box

The Conversion Box, which allows QMR and QMX spindles to be used together with PowerMACS 4000, have had to be configured for a particular spindle type using the tool ToolsTalk Service. From PowerMACS 4000 version 7.5.0 it is automatically configured by the PowerMACS system for the type of spindle that is specified by the setup.

The programming of the Box is handled by the TC and takes place when any of the following two situations occur:

- The setup loaded to the TC specifies another spindle article number than the one that the Box currently is configured for.
- The TC detects that the value of one or more of a number of crucial parameters differs between the Box and the PowerMACS setup (the calibration parameters are not included in the comparison).

The TC indicates that it has programmed the Conversion Box by issuing one of the following events depending on the outcome:

- 4234, Conv. Box: TC n, Box programmed successfully
- 4232, Conv. Box: TC n, Failed to program box with reference values from setup

Whenever the Box is programmed by the TC, or the TC detects that the Box has been replaced, it requires that a Motortune is run for the spindle before it allows a cycle to be started. This is indicated by the following events:

- 4231, Conv. Box: TC n, New box detected
- 4233, Conv. Box: TC n, Motortune is required and calibration is recommended

If Motortune is required the TC will remember that also if the power supply is cycled.

The only exception to this is the first time a TC connected to a correctly programmed Box is updated to version 7.4.2. If all important parameters have the same values as stated by the latest spindle type definition files, then Motortune is not required.

If the Box has been calibrated using an earlier TTPM version its checksum is most likely incorrect even though all values are correct. This problem is also checked for and corrected the first time the TC is upgraded to version 7.4.2 if a Box is connected at that time. When corrected the following warning event is issued:

- 4235, Conv. Box: TC n, Checksum faults detected. Verify spindle calibration values

Please note that even if the checksum is automatically corrected you should verify that the calibration values are correct.

Should a Box calibrated with an old TTPM version be connected to an upgraded TC at a later time then the following events will be issued:

- 5262, TC 1, Error reading reference from spindle. Ref 1, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 1, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 2, Status 50

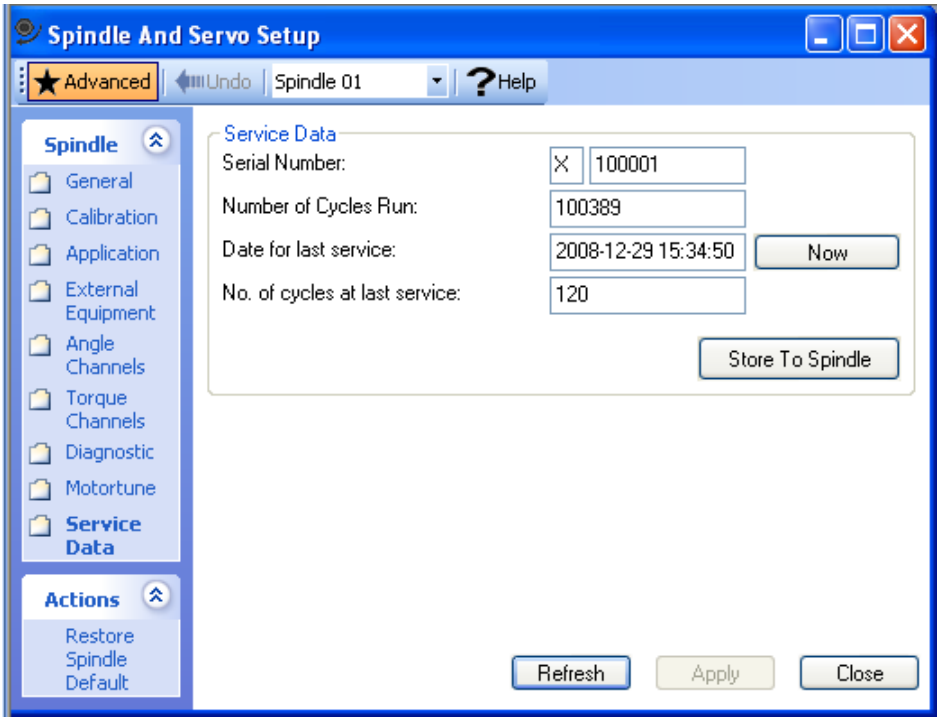
- 5262, TC 1, Error reading reference from spindle. Ref 3, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 4, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 11, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 32, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 60, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 61, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 62, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 63, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 64, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 65, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 89, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 90, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 100, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 109, Status 50
- 5262, TC 1, Error reading reference from spindle. Ref 170, Status 50
- 5262, TC 1, Error writing reference to spindle. Ref 62, Status 21

after which the Box is programmed with default values.

When the Box is programmed all parameters that are important for controlling the spindle, including the calibration parameters, are set. However, the following parameters are left unchanged:

- The serial number of the spindle
- The total number of cycles run
- The total number of cycles run at last service
- The date when the last service was made

The value of these parameters can be set, at any time, using the new "Service Data" page of the TTPM Spindle Set Up form when on-line to a spindle of Conversion Box type.



The configuration parameters for a spindle are defined in the spindle type file accompanying the spindle. From version 7.5.0 PowerMACS uses the same version of these files as ToolsTalk Service does (extension .TTT, instead of .TTS).

These latest version of these files are installed in the folder "SpindleType" which is located in the folder that TTPM was installed to (normally "C:\ PM4000 7.4.2"). By default this folder only contains files for QST spindles. The files corresponding to QMR and QMX spindles are located in subfolders named "QMR" and "QMX" respectively and must be copied to the "SpindleType" folder before they can be seen and used by TTPM.

TTPM will refuse to convert an old setup to version 7.4.2 unless it can find the spindle type files of all used spindles of QMR and QMX type.

11 References

11.1 References to external documents

1. “Functional and Technical Description PowerMACS 4000 TCs”
2. "Calibration with ACTA", Atlas Copco Tools document No. 9836 2086 00
3. “Open Protocol MID for PowerMACS data”

12 Glossary of Terms

12.1 Glossary

This part describes some basic concepts that are used throughout the complete document and have very specific meanings.

System

A *system* comprises one or more Tightening Controllers linked together. A system works autonomously, performing a tightening task.

Station

Within a system there is one or more *stations*. A station controls the tightening of one or more bolts. It has a PLC to control the tightening cycles.

Bolt

A *bolt* is the object that will be tightened by the system.

Spindle

A *spindle* performs the actual tightening of a bolt. It normally comprises:

- Motor
- Torque sensor
- Angle sensor

Often one bolt is tightened with one spindle, but it is possible to configure the system so several bolts are tightened with one spindle.

Program, steps and sequences

How the tightening of the bolt should be done is described in a tightening *program*. The program comprises *steps*, each describing how a part of the tightening cycle should be performed.

A number of steps can be saved as a *sequence*. When building a new tightening program, or altering an existing one, it is possible to insert sequences.

Mode

When a tightening cycle is started, the choice of tightening program is done using the *Mode* signal in the PLC. With use of a mode table it is defined which tightening program that should be used for each of the specific bolts in each mode. Each station in a system has its own unique *Mode* table.

Cycle

A tightening *cycle* is all operations from a start signal until ready to start next cycle.

Reject Management

Reject management is an alternative sequence to be followed, in the event of a NOK situation with the purpose to achieve an OK condition by doing a second try.

Shift

A shift is a time period. It is programmable when shifts starts and stops during a day (weekends may be different). Some results/reports are related to shifts.

Console Computer

The *console* or *console computer* is the PC computer used to set up a PowerMACS system. It is not needed for automatic running, but it can optionally be used to monitor the system and to collect and present various data using ToolsTalk PowerMACS.

One console computer can be used for several systems, if these are hooked up on the same computer network, but only one system at a time.

ToolsTalk PowerMACS

ToolsTalk PowerMACS is the graphical user interface of the PowerMACS 4000 system. It is a Windows based program that executes on the console computer (or any PC). It is used to set up a PowerMACS 4000 system and can be used to monitor the system and to collect and present various data.

Setup and Tables

A *setup* is a group of data that completely describes the system. The system uses the information in the *setup* to know what to do. A *setup* could also be called a *configuration*. A *setup* can be stored on the console computer in a file on a hard disk or on a floppy. It can be downloaded to the system, or uploaded for storing or alteration.

A *setup* is made up of *tables*, each containing data for a specific topic, e.g. tightening program, SPC *setup* etc.

Setups are manipulated with the ToolsTalk PowerMACS application program on the console computer. It is possible to export and import certain tables, but the setup itself is a complete package.

<system torque unit>

<system torque unit> is the torque unit selected for the system, or setup. This is selected using the Set Up Options form.

PLC

In every PowerMACS station there is a *PLC* (Programmable Logic Controller). The PLC is used as ”glue” between various functions within the system like:

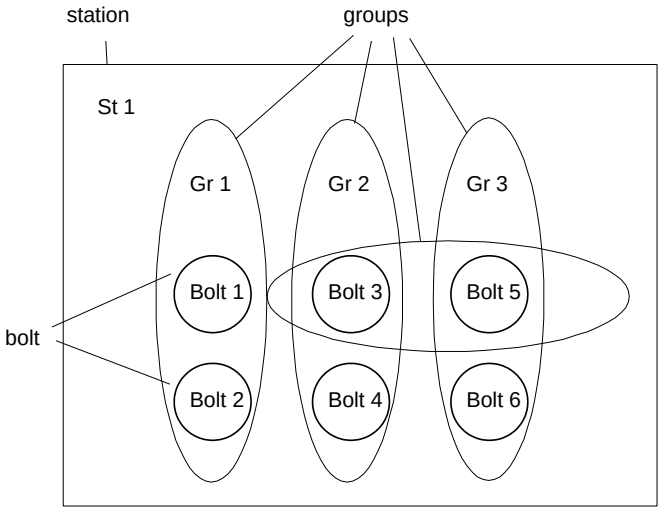
- Start of tightening from digital inputs
- Output of status to digital outputs
- Controlling flow of data
- Analyzing ID codes

Group

In many cases, when there is a problem with one bolt, reject actions must be performed not only on this bolt but with others within the station. In PowerMACS it is therefore possible to put bolts together in *groups*.

A group is a set of bolts, which are run completely separate from each other, but have some form of relationship. In case of reject management, they are considered to belong to the same group, and special actions can be specified for bolts within the group.

The bolts in a station can be divided into groups of bolts as in following figure:



All bolts belong to the station St 1. Bolts 1 and 2 belongs to group Gr 1, bolt 3 and 4 to group Gr 2, bolt 5 and 6 to group Gr 3. A special case is bolt 3 and 5, which both belongs to two groups at the same time.

9836 3521 01
2009-11
Edition 10.0.0

www.atlascopco.com