



## *User Guide*

---

# ***SM-Register***

---

Solutions Module for:

- Unidrive SP
- Digitax ST

Part Number: 0471-0117-02  
Issue: 2



**CONTROL  
TECHNIQUES**

[www.controltechniques.com](http://www.controltechniques.com)

## **General Information**

The manufacturer accepts no liability for any consequences resulting from inappropriate, negligent or incorrect installation or adjustment of the optional operating parameters of the equipment or from mismatching the variable speed drive (drive) with the motor.

The contents of this guide are believed to be correct at the time of printing. In the interests of a commitment to a policy of continuous development and improvement, the manufacturer reserves the right to change the specification of the product or its performance, or the contents of this guide, without notice.

All rights reserved. No parts of this guide may be reproduced or transmitted in any form or by any means, electrical or mechanical including photocopying, recording or by an information storage or retrieval system, without permission in writing from the publisher.

## **Drive software version**

This product is supplied with the latest version of user-interface and machine control software. If this product is to be used in a new or existing system with other drives, there may be some differences between their software and the software in this product. These differences may cause this product to function differently. This may also apply to drives returned from a Control Techniques Service Centre.

If there is any doubt, contact a Control Techniques Drive Centre.

---

# Contents

---

<b>1</b>	<b>Safety Information .....</b>	<b>6</b>
1.1	Warnings, Cautions and Notes .....	6
1.2	Electrical safety - general warning .....	6
1.3	System design and safety of personnel .....	6
1.4	Environmental limits .....	7
1.5	Compliance with regulations .....	7
1.6	Motor .....	7
1.7	Adjusting parameters .....	7
<b>2</b>	<b>Introduction .....</b>	<b>8</b>
2.1	Features for SM-Register .....	8
2.2	SM-Register identification .....	9
2.3	Conventions used in this guide .....	9
2.4	PC development software .....	9
2.5	User knowledge .....	10
<b>3</b>	<b>Installation .....</b>	<b>11</b>
3.1	General installation .....	11
3.2	SMARTCARD installation .....	11
3.3	Electrical connections .....	11
3.4	Connections .....	13
3.5	CTNet cable .....	13
3.6	CTNet network termination .....	14
3.7	EIA-485 connections .....	15
3.8	Digital I/O connections .....	17
3.9	Port isolation .....	17
<b>4</b>	<b>Getting started .....</b>	<b>18</b>
4.1	Using SyPTPro .....	18
4.2	Connecting the PC to the SM-Register .....	18
4.3	CTNetAPI routing .....	19
4.4	Configuring communications within SyPTPro .....	19
4.5	Creating a node in SyPTPro .....	19
4.6	DPL programming basics .....	20
4.7	Program example .....	21
4.8	Downloading programs .....	22
<b>5</b>	<b>SM-Register Functionality .....</b>	<b>23</b>
5.1	Introduction to SM-Register .....	23
5.2	Registration sensor connections .....	26
5.3	Functional description .....	26
5.4	Registration function blocks .....	45

<b>6</b>	<b>Parameters .....</b>	<b>67</b>
6.1	Overview .....	67
6.2	Saving parameters .....	67
6.3	Configuration parameters .....	68
6.4	Menus 70-75 - PLC registers .....	79
6.5	Menu 85 - Timer function parameters .....	80
6.6	Menu 86 - Digital I/O parameters .....	82
6.7	Menu 88 - Status parameters .....	83
6.8	Menu 90 - General parameters .....	85
6.9	Menu 91 - Fast access parameters .....	92
6.10	Menus 18,19 - Application parameters .....	97
6.11	Menu 20 - Application menu .....	98
<b>7</b>	<b>DPL Programming .....</b>	<b>99</b>
7.1	Program header .....	99
7.2	Tasks .....	100
7.3	Variables .....	102
7.4	Parameters .....	104
7.5	Operators .....	105
7.6	Basic DPL commands .....	106
7.7	User defined function blocks .....	111
<b>8</b>	<b>Communications .....</b>	<b>113</b>
8.1	EIA-485 Serial communications port .....	113
8.2	CTNet .....	119
8.3	SM-Register mapping parameters (fieldbus) .....	120
<b>9</b>	<b>Freeze and Marker .....</b>	<b>122</b>
9.1	Freeze input .....	122
9.2	Marker pulse .....	123
<b>10</b>	<b>CTSync .....</b>	<b>124</b>
10.1	Overview .....	124
10.2	Connections .....	124
10.3	Limitations .....	125
10.4	CTSync function blocks .....	125
10.5	Motion engine .....	126
10.6	Virtual master example .....	129
<b>11</b>	<b>Inter-option synchronisation .....</b>	<b>131</b>
11.1	Overview .....	131
11.2	Inter-Option synchronisation example 1 .....	131
11.3	Inter-option synchronisation example 2 .....	134
11.4	Position control tasks .....	138

<b>12</b>	<b>Diagnostics .....</b>	<b>139</b>
12.1	Run-time errors .....	139
12.2	Drive display trip codes .....	139
12.3	SM-Register run-time error codes .....	140
12.4	Handling run-time errors with the ERROR task .....	142
12.5	Resource monitoring .....	143
12.6	Support .....	144
<b>13</b>	<b>Quick Reference .....</b>	<b>145</b>
	<b>Index .....</b>	<b>150</b>

---

# 1 Safety Information

---

## 1.1 Warnings, Cautions and Notes



A **Warning** contains information, which is essential for avoiding a safety hazard.



A **Caution** contains information, which is necessary for avoiding a risk of damage to the product or other equipment.



A **Note** contains information, which helps to ensure correct operation of the product.

## 1.2 Electrical safety - general warning

The voltages used in the drive can cause severe electrical shock and/or burns, and could be lethal. Extreme care is necessary at all times when working with or adjacent to the drive. Specific warnings are given at the relevant places in this User Guide.

## 1.3 System design and safety of personnel

The drive is intended as a component for professional incorporation into complete equipment or a system. If installed incorrectly, the drive may present a safety hazard.

The drive uses high voltages and currents, carries a high level of stored electrical energy, and is used to control equipment which can cause injury.

Close attention is required to the electrical installation and the system design to avoid hazards either in normal operation or in the event of equipment malfunction. System design, installation, commissioning and maintenance must be carried out by personnel who have the necessary training and experience. They must read this safety information and this User Guide carefully.

The STOP and SECURE DISABLE / SAFE TORQUE OFF functions of the drive do not isolate dangerous voltages from the output of the drive or from any external option unit. The supply must be disconnected by an approved electrical isolation device before gaining access to the electrical connections.

**With the sole exception of the SECURE DISABLE / SAFE TORQUE OFF function, none of the drive functions must be used to ensure safety of personnel, i.e. they must not be used for safety-related functions.**

Careful consideration must be given to the functions of the drive which might result in a hazard, either through their intended behaviour or through incorrect operation due to a fault. In any application where a malfunction of the drive or its control system could lead to or allow damage, loss or injury, a risk analysis must be carried out, and where necessary, further measures taken to reduce the risk - for example, an over-speed protection device in case of failure of the speed control, or a fail-safe mechanical brake in case of loss of motor braking.

The SECURE DISABLE / SAFE TORQUE OFF function has been indepentently approved approved by BGIA as meeting the requirements of EN954-1 category 3 for the prevention of unexpected starting of the drive. It may be used in a safety-related application. **The system designer is responsible for ensuring that the complete system is safe and designed correctly according to the relevant safety standards.**

### 1.4 Environmental limits

Instructions in the drive *User Guide* or *Technical Data Guide* regarding transport, storage, installation and use of the drive must be complied with, including the specified environmental limits. Drives must not be subjected to excessive physical force.

### 1.5 Compliance with regulations

The installer is responsible for complying with all relevant regulations, such as national wiring regulations, accident prevention regulations and electromagnetic compatibility (EMC) regulations. Particular attention must be given to the cross-sectional areas of conductors, the selection of fuses or other protection, and protective ground connections.

The drive *User Guide* or *Technical Data Guide* contains instructions for achieving compliance with specific EMC standards.

Within the European Union, all machinery in which this product is used must comply with the following directives:

98/37/EC: Safety of machinery.

89/336/EEC: Electromagnetic Compatibility.

### 1.6 Motor

Ensure the motor is installed in accordance with the manufacturer's recommendations. Ensure the motor shaft is not exposed.

Standard squirrel cage induction motors are designed for single speed operation. If it is intended to use the capability of the drive to run a motor at speeds above its designed maximum, it is strongly recommended that the manufacturer is consulted first.

Low speeds may cause the motor to overheat because the cooling fan becomes less effective. The motor should be installed with a protection thermistor. If necessary, an electric forced vent fan should be used.

The values of the motor parameters set in the drive affect the protection of the motor. The default values in the drive should not be relied upon.

It is essential that the correct value is entered in Pr **0.46** motor rated current. This affects the thermal protection of the motor.

### 1.7 Adjusting parameters

Some parameters have a profound effect on the operation of the drive. They must not be altered without careful consideration of the impact on the controlled system. Measures must be taken to prevent unwanted changes due to error or tampering.

---

## 2 Introduction

---

### 2.1 Features for SM-Register

Modern variable speed drives offer a multitude of in-built features such as ramp control, PID loops, simple position control, etc. However this functionality is limited. The drive can only do so many things and when it comes to controlling more complex applications, users often have to resort to using external equipment such as PLCs to control the drive from a system point of view.

However the flexibility of certain drives can be substantially increased by using the SM-Register. The SM-Register provides an additional processor for the drive and allows the user to utilise existing, or write their own, application-specific software. They also offer powerful networking capabilities so many drives (and other equipment) can be connected together to communicate process wide information thus offering a complete application solution. The SM-Register is a solutions module that can be installed to expansion slots in the supported drives and is powered from the drive's internal power supply.

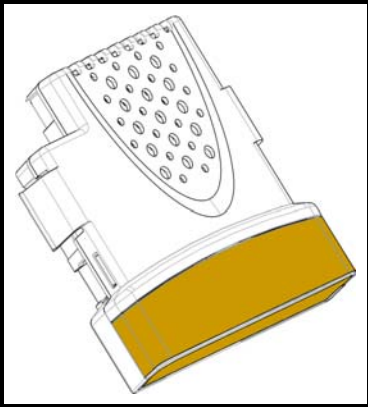
#### Specifications for SM-Register

- Registration position capture functionality.
- Enhanced high speed dedicated microprocessor.
- 384kb Flash memory for user program
- 80kb user program memory.
- EIA-485 port offering ANSI, Modbus-RTU slave and master and Modbus-ASCII slave and master protocols.
- CNet high speed network connection offering up to 5Mbit/s data rate.
- Two 24V digital inputs.
- Two 24V digital outputs.
- Task based programming system allowing for real-time control of drive and process.
- CTSync.



## 2.2 SM-Register identification

Figure 2-1 SM-Register



The SM-Register can be identified by:

1. The label located on the underside of the module.
2. The color coding across the front of the SM-Register.  
SM-Register: Green Brown.

## 2.3 Conventions used in this guide

The configuration of the host drive and SM-Register module is done using menus and parameters. A menu is a logical collection of parameters that have similar functionality. In the case of a SM-Register, the parameters will appear in menu 15, 16 or 17 for the drive depending on the slot the module is installed into. The menu is determined by the number before the decimal point.

The method used to determine the menu or parameter is as follows:

- Pr **xx.00** - signifies any menu and parameter number 00.
- Pr **mm.XX** - where **mm** signifies the the menu allocated to the SM-Register (this could be 15, 16 or 17) and **XX** signifies the parameter number.

## 2.4 PC development software

Application programs for the SM-Register may be developed by the user with the SyPT software tools.

SyPTPro offers various tools to help in developing solutions:

- Configuration editor for configuring drives and connections on CTNet, EtherNet, CT-RTU, CT-TCP and MD29MON networks.
- IEC61131-3 based ladder and function block programming.
- Native DPL language programming.
- Watch window for monitoring drive / option parameters, and program variables.
- Single-stepping and breakpoint debugging facilities.

With SyPTPro you may connect to the SM-Register by either:

- Direct connection to the EIA-485 port on the front of the drive.
- Connecting to one or more options on a CNet network (a CNet interface card for the PC will be required).
- Ethernet.

SyPTPro runs under Microsoft Windows™ 2000/XP/Vista 32 bit.

## 2.5 User knowledge

When developing custom applications using SM-Register it is beneficial to have an understanding of real-time task and event driven programming. A good appreciation of a high level programming language such as BASIC or C is advantageous. For people with a previous experience of programming in Ladder Logic, SM-Register offers Ladder Diagrams and Function Block diagrams (FBD).

This *User Guide* assumes the user has at least superficial knowledge of Microsoft Windows™.

# 3 Installation

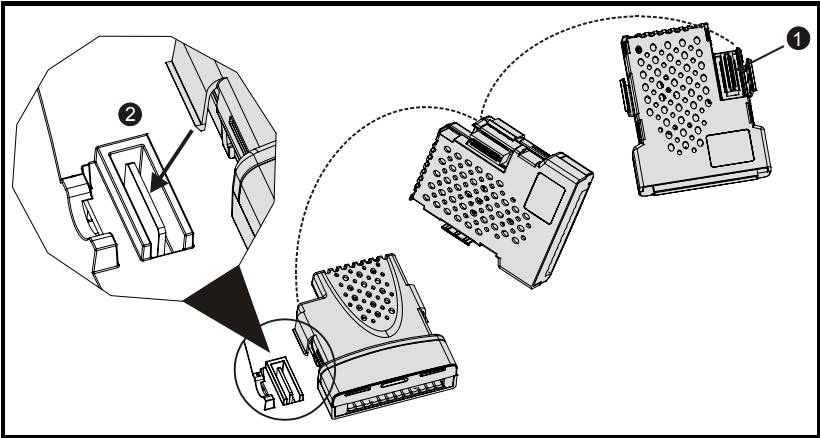


Before installing or removing an SM-Register module in any drive, ensure the AC supply has been disconnected for at least 10 minutes and refer to Chapter 1 *Safety Information* on page 6. If using a DC bus supply ensure this is fully discharged before working on any drive or Solutions Module.

## 3.1 General installation

The installation of a SM-Register module is illustrated in Figure 3-1.

**Figure 3-1 Fitting a SM-Register module**



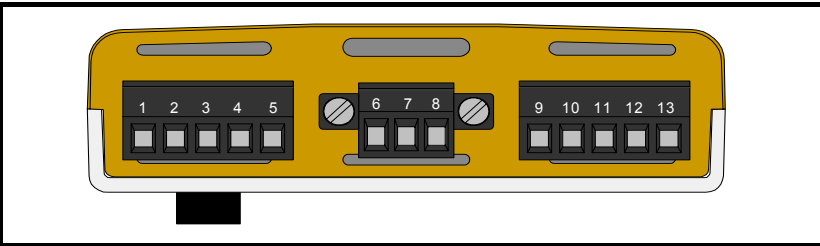
The SM-Register module connector is located on the underside of the module (1). Push this into the Solutions Module slot located on the drive until it clicks into place (2). Note that some drives require a protective tab to be removed from the Solutions Module slot. For further information, refer to the appropriate drive manual.

## 3.2 SMARTCARD installation

Refer to the drive *User Guide* for SMARTCARD support.

## 3.3 Electrical connections

**Figure 3-2 SM-Register - Front view**



The terminal functions are given in Table 3.1.

**NOTE** Ensure that the drive is off before removing any modules.

**Table 3.1 Module connectors**

Terminal	Function	Description
1	0V SC	0V connection for EIA-485 port
2	/RX	EIA-485 Receive line (negative) Incoming.
3	RX	EIA-485 Receive line (positive) Incoming.
4	/TX	EIA-485 Transmit line (negative) Outgoing.
5	TX	EIA-485 Transmit line (positive) Outgoing.
6	CTNet A	CTNet data line.
7	CTNet Shield	Shield connection for CTNet.
8	CTNet B	CTNet data line.
9	0V	0V connection for digital I/O.
10	DIO	Digital input 0.
11	DI1	Digital input 1.
12	DO0	Digital output 0.
13	DO1	Digital output 1.

**Table 3.2 Digital input specifications**

<b>Terminal 10 / Digital Input 0</b>	
<b>Terminal 11 / Digital Input 1</b>	
Type	Positive logic IEC 61131-2
Maximum Input Voltage	+/- 30V
Switching Threshold	9.5V +/- 0.3V
Load	2mA at +15V

**Table 3.3 Digital output specifications**

<b>Terminal 12 / Digital Output 0</b>	
<b>Terminal 13 / Digital Output 1</b>	
Type	Positive logic when active
Output Voltage	0 / 24V
Nominal max. output	20mA

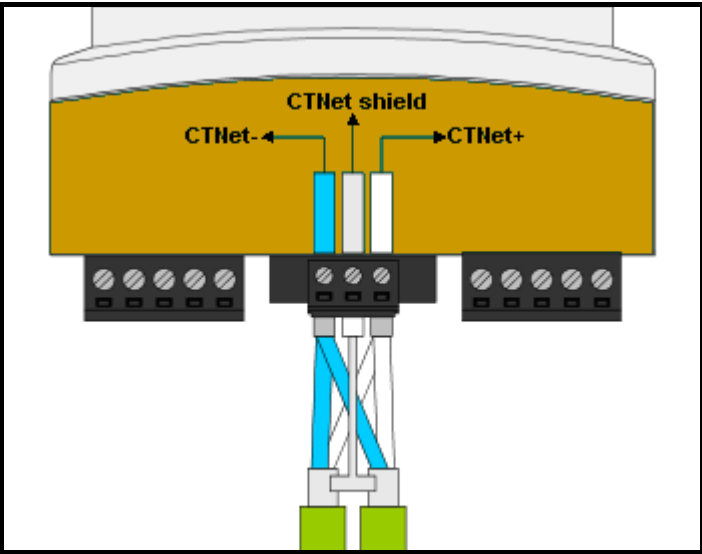
**NOTE** When inactive Digital Outputs are effectively floating.

# 3.4 Connections

**NOTE** This *User Guide* covers only the basics of connecting a CTNet network. Please refer to the CTNet *User Guide* for full information.

To connect the module to the CTNet network, make the connections as shown in the diagram below.

**Figure 3-3 CTNet network connections**



The shields of both cables should be twisted together and connected into the centre terminal of the 3-way terminal block. This “pigtail” should be kept as short as possible.

# 3.5 CTNet cable

CTNet cable has a single twisted pair plus overall shielding. One set of data terminals is provided. This has the advantage that if the terminal block is unplugged the continuity of the CTNet network is not broken.

CTNet networks run at high data rates, and require cable specifically designed to carry high frequency signals. Low quality cable will attenuate the signals, and may render the signal unreadable for the other nodes on the network. The only approved CTNet cable is that supplied by Control Techniques.

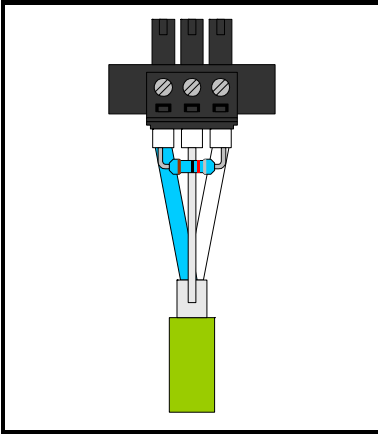
Safety Information
Introduction
Installation
Getting started
SM-Register Functionality
Parameters
DPL Programming
Communications
Freeze and Marker
CTSync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

### 3.6 CTNet network termination

It is very important in high-speed communications networks that the network communications cable is installed with the specified termination resistor network at each end of the cable. This prevents signals from being reflected back down the cable and causing interference.

The termination resistance should match as closely as possible the impedance of the cable. For the recommended Control Techniques green CTNet cable, an  $82\Omega$  0.25W termination resistor should be installed across the CTNet+ and CTNet- data lines at BOTH ends of the network segment.

**Figure 3-4 CTNet network termination**



Failure to terminate a network correctly can seriously affect the operation of the network. If the correct termination resistors are not installed, the noise immunity of the network is greatly reduced.

**NOTE**

If too many termination resistors are installed on a CTNet network, the network will be over-loaded, resulting in reduced signal levels.

### 3.6.1 CTNet cable shield connections

The cable shields should be linked together at the point where they emerge from the cable, and formed into a short pigtail to be connected to CTNet shield connection on the terminal block, as already shown.

For safety, the CTNet shield must be connected to ground at at least one point. This ground connection is to prevent the cable shield from becoming live in the event of catastrophic failure of another device on the CTNet network or along the cable run.

### 3.6.2 Maximum network length and number of nodes

The maximum number of nodes that can be connected to a single CTNet network is 255, however a network may need to be split into segments, separated by repeaters. The maximum length of network cable for a CTNet network is dependent on the baud rate and number of nodes. Refer to CTNet user guide for more information.

Repeaters can be purchased from Control Techniques. The table below shows the part numbers for the different repeaters available.

Part Number	Description
4500-0033	AI3-485X - Rev A
4500-0083	AI3-485X-CT - Rev D (Port 1), Rev A (Ports 2 and 3)
4500-0082	AI3-CT - Rev D
4500-0032	AI2-485X/FOG-ST - Rev A (Fibre Optic)
4500-0081	AI2-CT/FOG-ST - Rev D (Fibre-Optic)

Please refer to the CTNet *User Guide* for further guidance.

## 3.7 EIA-485 connections

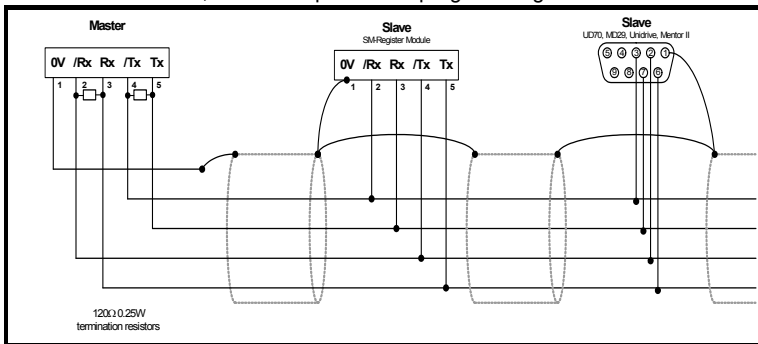
The EIA-485 port is available for lower-speed communications (up to 115,200bit/s). As standard the port supports the CT-ANSI slave, Modbus-RTU master / slave, and Modbus-ASCII master / slave protocols. Both 2 and 4-wire connections are possible.

More information on the use of the EIA-485 port can be found in Chapter 8 *Communications* on page 113.

A host controller can operate up to thirty-two EIA-485 devices with the use of line repeaters. Each transmitter and receiver of Control Techniques devices loads the line by 2 unit loads. Therefore in two-wire mode, each Control Techniques device loads the line by 4 unit-loads. This means that no more than a total of seven devices can be connected in a single group, allowing up to 4 unit-loads for the line repeater. Up to 15 devices can be connected if four-wire mode is used.

### 3.7.1 4 Wire EIA-485 network

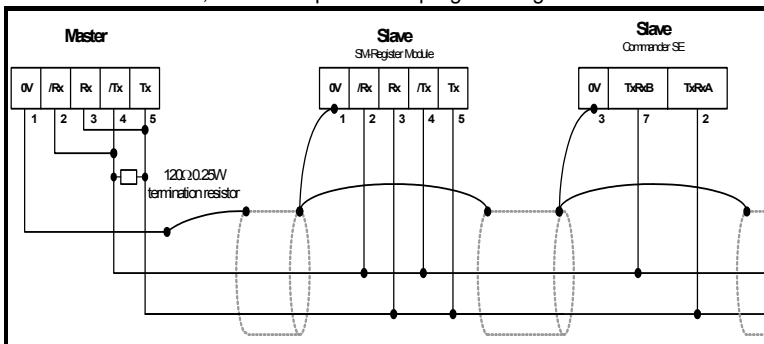
The diagram below shows the connections required for a 4 wire EIA-485 network, using a master controller with an EIA-485 port. SM-Register modules can be configured to act as master controllers, but this requires DPL programming to control the network.



An EIA-232 to EIA-485 converter is required to allow a standard PC serial port to communicate with a 4 wire EIA-485 network.

### 3.7.2 2 Wire EIA-485 network

The diagram below shows the connections required for a 2 wire EIA-485 network, using a master controller with an EIA-485 port. SM-Register modules can be configured to act as master controllers, but this requires DPL programming to control the network.



An EIA-232 to EIA-485 converter with “intelligent transceiver switching” (also known as “magic” EIA-485 converters) is required to allow a standard PC serial port to communicate with a 2 wire EIA-485 network. An example of a “magic” converter is the MA485F converter from Amplicon.

**NOTE** A “magic” converter is not required if the master controller has an RTS control output. This output is enabled when the master is transmitting, and disabled when the master is not transmitting. Control Techniques software packages do NOT switch the RTS line.

### 3.7.3 Grounding

It is recommended for safety that the shield of the communications cable be connected by a low-inductance path to a ‘clean’ ground point. This must only be done at one point.



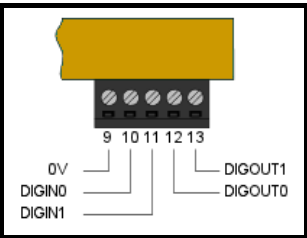
### 3.7.4 Routing of the cable

A data communications cable should not run parallel to any power cables, especially ones that connect drives to motors. If parallel runs are unavoidable, ensure a minimum spacing of 300mm (1 foot) between the communications cable and the power cable. Cables crossing one another at right-angles are unlikely to give trouble. The maximum cable length for a EIA-485 link is 1200 metres (4,000 feet). This is at low baud rates only. The higher the baud rate the lower the maximum cable length.

### 3.7.5 Termination

When a long-distance multi-drop EIA-485 system is used, the transmit and receive pairs should have a termination resistor of 120Ω installed across them in order to reduce signal reflections. However, at the lower data rates this is not so critical.

## 3.8 Digital I/O connections



The SM-Register is equipped with 2 digital inputs, known as DIGIN0 and DIGIN1, and 2 digital outputs, known as DIGOUT0 and DIGOUT1. These inputs and outputs can be read or written from the user program loaded into the SM-Register.

The digital outputs are a positive logic arrangement such that they are at +24V when active and will supply up to 20mA of current. When inactive they are effectively floating. The digital outputs are protected against short-circuit or overload. The trip threshold is 20mA, and if tripped both outputs will be deactivated.

The digital I/O are controlled using menu 86 - refer to section 6.6 *Menu 86 - Digital I/O parameters* on page 82.

## 3.9 Port isolation

The digital input/output ports are connected to the main drive control circuits.

The I/O circuits are isolated from the power circuits by basic insulation (single insulation) only. The installer must ensure that external control circuits are insulated from human contact by at least one layer of insulation (supplementary insulation) rated for the AC supply voltage.

The CTNet and EIA-485 ports have supplementary insulation from the input/output ports, giving overall double insulation from the power circuit. They have simple separation (functional insulation) from each other.

To maintain the double insulation status of the data ports:

- All circuits to which either port is connected must have protective separation (i.e. double insulation or single insulation with grounding).
- All circuits to which the drive control circuits are connected must have at least basic insulation from live parts.

---

## 4 Getting started

---

This chapter describes the basics of a user program using the SM-Register and some of the aspects of using SyPTPro.

Within the SM-Register the current slot menu is aliased as menu 81. Therefore when connected to the module via a communications link or from the user program, it is easiest if configuration parameters are referenced as menu 81.

Throughout the remainder of this User Guide, when referring to a specific parameter for any slot the format **Pr 81.XX** will be used. e.g. The *Autorun* parameter will be referred to as **Pr 81.13**.

This is also an aid to portability, as SM-Register modules with code using menu 81 can be installed to any slot, and the code should run as normal.

When the SM-Register modules are installed, the module identification parameter **Pr 81.01** will show 306.

The combination of parameters **Pr 81.02** and **Pr 81.51** provides the firmware version of the module. This User Guide is written for firmware version V01.01.00.

### 4.1 Using SyPTPro

SyPTPro provides the development platform for the SM-Register Module.

### 4.2 Connecting the PC to the SM-Register

There are three methods of connecting the programming PC to the SM-Register and these are outlined below:

#### 4.2.1 CTNet

With a CTNet connection you may connect the PC to a network of drives thereby allowing you to program and control all the drives directly from the PC. However you will need to have a CTNet interface card in your PC. There are PCI and USB cards available from Control Techniques for desktop and laptop computers.

Refer to section 3.4 *Connections* on page 13 for details of the CTNet connections on the SM-Register Module.

#### 4.2.2 EIA-485 serial port

You can connect the PC to the RJ45 serial port on the front of the drive. Special pre-made leads are available from Control Techniques for this purpose. These leads are to connect from your pc by either EIA-232 to EIA-485 or USB to EIA-485 - these leads are used with other Control Techniques products that use a RJ45 EIA-485 connector such as the Commander SE and Commander SK.

Please refer to your drive user guide for the positioning and pin-out descriptions of the RJ45 connector.

#### 4.2.3 Ethernet

With an SM-Ethernet module installed to the drive, you can use the CT-TCP/IP protocol to communicate to the drive using the supported PC tools such as SyPT Pro.

For more details on setting up Ethernet please refer to the SM-Ethernet user guide.

**Figure 4-1 Communications cable**



A number of drives may be connected via their EIA-485 ports on an RTU network, thus allowing the user to control any of those drives. In addition, if a drive has an SM-Register Module installed these will be seen by SyPT Pro along with any other SM-Register Module connected to this over CTNet. This is part of the routing capability of the CTNetAPI. See section 4.3 *CTNetAPI routing* on page 19.

### 4.3 CTNetAPI routing

The CTNetAPI offers a routing capability that allows access to various drives or SM-Register Module in a system. The user is able to download to, and upload from, these from within SyPTPro.

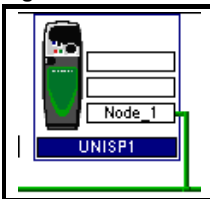
### 4.4 Configuring communications within SyPTPro

Before attempting to go *on-line* to the SM-Register Module you must set SyPTPro up to use the correct communications protocol:

1. In the SyPTPro Configuration Editor select **PC Communications Settings** from the Run menu
2. If connecting via CTNet, select CTNet as the protocol and make sure the baud rate is correct (pressing the Help button will show details of the other settings).
3. If connecting via RS232/485 to the front of the drive select **CT-RTU** as the protocol and choose the appropriate RS232 COM port. Also ensure drive parameters **Pr 11.24** and **Pr 11.25** are set to **rtU** and **19200** respectively (these are the default values).
4. Press OK.

### 4.5 Creating a node in SyPTPro

**Figure 4-2 Node icon**



1. Insert a new node by selecting **Node** from the Insert menu or by double-clicking on the **Double-click to Insert Node** icon.
2. The **Node** properties will now be displayed. Enter in the relevant details for Node ID and Network. Then any information on Solutions Modules can be entered using the tabs at the top of the box.
3. Press OK.

## 4.6 DPL programming basics

The SM-Register Module can be programmed using a mixture of ladder diagrams (LD), function block diagrams (FBD) and DPL code (Drive Programming Language). Collectively they are known as a *DPL Program*. Only the ladder programming is available with SyPTLite.

At the very top level a program consists of:

- Program Header - giving the program title, author, version, etc. This is configured using the node properties dialogue box in SyPTPro.
- Program Body - comprised of *task* sections containing LD, FBD and DPL sections. This is created in the DPL Editor within SyPTPro.

Task sections encapsulate blocks of instructions that are to be executed by the microprocessor at a particular time, for example every 8ms or when the module first powers-up. Each task has a particular name, purpose and priority. Refer to section 7.2 *Tasks* on page 100 for further information.

### 4.6.1 Function block library

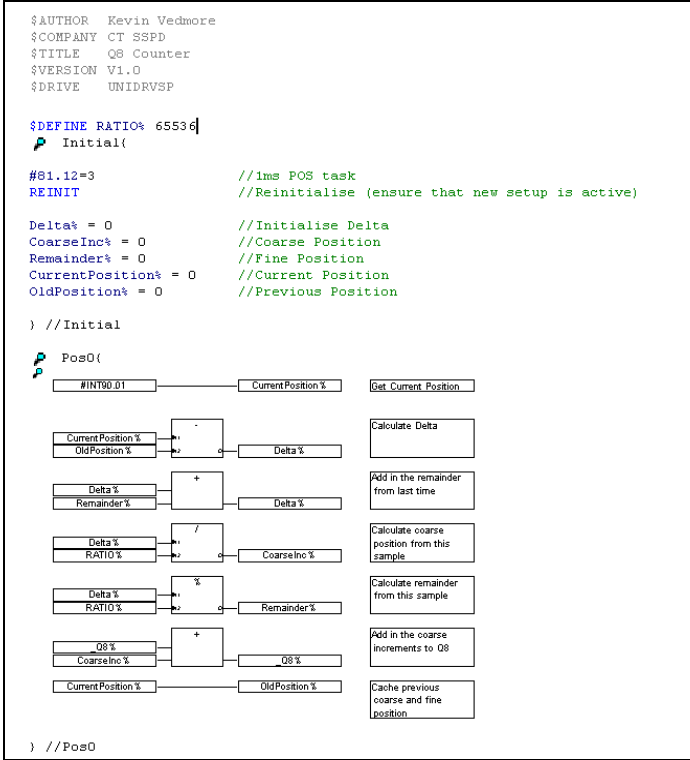
SyPTPro and SyPTLite come with an extensive library of pre-made function blocks. These perform tasks from simple things like a counter to more complex things such as PID loops or S-Ramp profile generators. These pre-supplied blocks are known collectively as the Function Block Library (FBL).

The functions in the FBL are documented in the on-line help.

You can also create your own function blocks within your program. So if you've created a new profile generator, you may encapsulate it within a user-defined function block (UDFB) and incorporate into your main DPL program. See section 7.7 *User defined function blocks* on page 111 and the on-line help for information.

Shown below is an example of a DPL program written within SyPTPro:

Figure 4-3 Example DPL program



This program will take the positional feedback information from the drive (which is scaled to  $2^{32}/\text{rev}$ ), work out the delta (which will be proportional to speed) and convert to encoder counts (based on a standard quadrature encoder) and add this to an accumulator.

This example shows the basic concepts of accessing parameters and using mathematical functions. It may be useful to people migrating from the UD70 platform toj Unidrive Classic since it will show how to re-create the `_Q8%` accumulative encoder position value as was available on that product.

Running through the program there are four distinct sections:

- Header section
- An **Initial** task
- A **Pos0** task
- A Function block diagram

4.7.1 Header section

This section is automatically generated by SyPTPro from the details in the node properties dialogue box. It contains information such as the program's title, author and version.

Safety Information
Introduction
Installation
Getting started
SM-Register Functionality
Parameters
DPL Programming
Communications
Freeze and Marker
CTSync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

### 4.7.2 Initial task

As explained later in section 7.2 *Tasks* on page 100, this is a *task* which is executed when the SM-Register Module is first powered up or is reset, providing the Autorun parameter is set (refer to the section on saving parameters). In this task are some DPL statements that initialise some integer variables (denoted by a trailing % symbol) to zero.

### 4.7.3 Pos0 task and the function block diagram

Because this program will be dealing with position feedback information, the bulk of the work will be done in the POS0 task. Any operations involving speed, position or torque control are usually done in the POS0, and POS1 or the CLOCK task which is now synchronised to the drive. In this case there is a single function block diagram which does all the calculations we need to work out the incremental encoder position.

The basic steps taken are:

1. Read the current encoder feedback value.
2. Subtract the previously read encoder feedback value to give us the delta.
3. Re-scale the value to actual encoder counts, assuming a standard incremental (rather than SinCos type) encoder.
4. Add this delta to an accumulator.
5. Remember the current encoder position for next time.

In this example program a variable, `_Q8%`, is used. This is a 32-bit value just like any other variable, but is part of a special set of *registers* known as the PLC register set. These PLC registers have the advantage of being able to be saved into non-volatile memory and also are accessible via parameters in menu 70 through to 76. More information on these can be found in section 6.4 *Menus 70-75 - PLC registers* on page 79.

#### NOTE

If you wish to create and try this program yourself and you have not used SyPTPro software before then it is advised to read through the remainder of this chapter first, then read the **Getting started** section of *SyPTPro Help* which explains how to create such a program.

In order to ensure that the POS0 task is executed, parameter **Pr 81.12** must be set to a non-zero value in the Initial task. After setting this, a REINIT command must be issued (see below).

```
#81.12 = 3 //Pos task schedule period 1ms
REINIT    //Reinitialise
```

## 4.8 Downloading programs

By default, programs can only be downloaded to the SM-Register Module when the drive enable signal is not active (**Pr 6.15**=0). This behaviour can be disabled by setting **Pr 81.37** to 0.

Safety Information
Introduction
Installation
Getting started
<b>SM-Register Functionality</b>
Parameters
DP-PL Programming
Communications
Freeze and Marker
CT-Sync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

# 5 SM-Register Functionality

## 5.1 Introduction to SM-Register

The SM-Register module offers all of the functionality of an SM-Applications Plus module, with additional registration position capture and event filtering capabilities, enabling the user to develop a wide range of registering systems, from simple position measurement to pattern recognition.

### 5.1.1 Application uses for the SM-Register module

The flexibility of the SM-Register module makes it ideal for any application which requires an advance / retard calculation, or a position / distance measurement to be taken. The following list gives some examples of the type of applications the SM-Register module may be used for:

- Rotary / linear printing.
- Rotary Knife / Shear.
- Flying Shear.
- Labelling machines.
- Sorting machines / conveyors.
- Product finishing machines e.g. edge cutting / trimming.
- Packaging machines.

### 5.1.2 Features

The SM-Register module has the following additional features:

- 2 fully independent capture channels.
- Edge capture - captures the position of negative and / or positive edges.
- Pulse capture - captures the position and width of both positive or negative pulses.
- Pulse width filtering.
- Edge filtering (distance from previous edge).
- Registration windowing, including automatic window advance / retard.
- Automated pattern recognition. Patterns of pulses and gaps may be detected, in both positive and negative pulse capture modes.
- Sensor signal rise / fall time compensation, for high speed applications.
- Administration tools for calculating positions and distances.

### 5.1.3 Advantages over standard freeze

The Freeze scheme present on the SM-Applications range of modules, and the SM-Register module, has similar but far more limited capturing and data manipulation capabilities to those present in the SM-Register position capture system.

The Freeze scheme allows the user to latch the Feedback and Reference encoder positions, of a single positive or negative edge, at a maximum rate of once every 3 position update cycles, and must be manually reset in order to capture a new position. The user can not switch between negative and positive edge capture on the fly, therefore pulse capturing is not possible.

The SM-Register module has the ability to repeatedly capture the reference or feedback encoder position, but at a much higher rate than the Freeze scheme:

**Table 5.1 Burst edge capture for SM-Register**

<b>Burst Edge Capture For SM-Register (Single channel active)</b>	
Position Update Rate	Maximum Number Of Edges
250µs	8
500µs	10
1ms	25
2ms	50
4ms	100
8ms	200

The edges generated for the burst data test were nominally 12µs apart.

**Table 5.2 Sustained edge capture for SM-Register**

<b>Sustained Edge Capture For SM-Register (Single channel active)</b>	
Position Update Rate	Microseconds Per Edge
250µs	80
500µs	70
1ms	60
2ms	60
4ms	60
8ms	60

**NOTE** The data above represents the peak expected values while using the SM-Register system; the actual throughput in a real application may be lower where large volumes of additional code are also being handled by the SM-Register, e.g. motion processing or system sequencing. This is because each event received uses processor resources to handle the data, therefore any other operation which uses additional resources will reduce the peak burst and sustained pulse handling performance.

If both channels are active the above event rates are approximately halved.

From Table 5.1 and Table 5.2 it can be seen that for a nominal position update rate of 1ms, the SM-Register capture will:

- Handle a burst of 72 edges in the same amount of time it would take the Freeze system to handle 1 edge.
- Handle a sustained throughput of 48 edges in the same amount of time it would take the Freeze system to handle 1 edge.

The SM-Register capturing system can handle not only positive edges or negative edges, but negative and positive edges at the same time, negative or positive pulses (providing leading edge position pulse width and trailing edge position information), and patterns of pulses.

When repeatedly capturing events the SM-Register module scheme is automatically reset by the operating system, unlike the Freeze system which must be manually reset which takes much longer; this ability facilitates the high event throughput of the SM-Register system.

Without additional user code the Freeze system is not capable of filtering event data to



reject “noise”, windowing unwanted events out (in hardware), or compensating for sensor rise and fall times at high speed, whereas these are standard features of the SM-Register system and only need to be configured to suit the application.

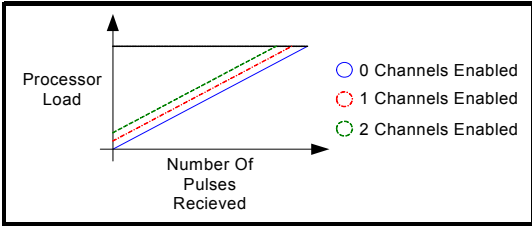
**NOTE** The SM-Register and Freeze capturing systems are entirely separate entities, and can both be used at the same time on SM-Register.

### 5.1.4 Considerations when using SM-Register

The following list highlights some items which must be considered when creating a registration system using the SM-Register module:

- Processing resource used is proportional to the number of pulses the SM-Register system receives, since every pulse received must be handled. It is therefore possible that if an excessively large number of pulses are received, that the SM-Register module will completely run out of processing resource, causing a SLx. Er (TR54 - DPL Task Overrun) trip which will stop the system. To prevent the system from receiving unwanted pulses, the windowing scheme may be employed to reject unwanted events; it does this by turning on and off capture such that only events within the window will be received by the system, and therefore consume resource.

Figure 5-1 Processor usage vs pulses received

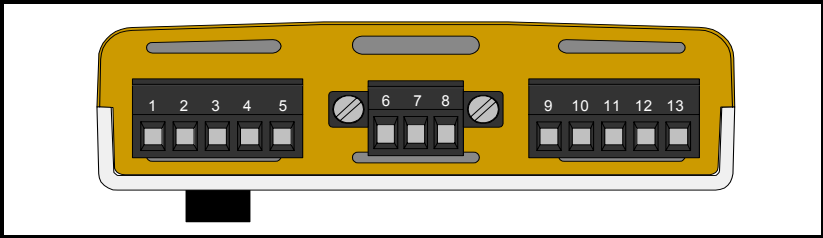


- The SM-Register module is capable of everything that an SM-Applications Plus module can do, however it is recommended that the SM-Register module is only used to perform the registration position capture and offset calculation, and that an additional SM-Applications module is used to handle any motion required to implement the resulting registration offset. However, if the system has a small volume of additional application code, and the number of pulses received will be low the SM-Register module may be able to handle the complete application; the user is responsible for testing their system to verify this.
- The SM-Register capture functionality and the Menu 85 timers function are mutually exclusive, therefore when the SM-Register is scheme is enabled the timer will not be available to the user.
- The method of determining the captured event positions within the SM-Register module operates in a similar way to the Freeze system when a SinCos encoder is used, and will provide similar results. However, the system used with an incremental encoder Freeze is entirely different and would result in a worst case difference of 3 motor encoder counts based on a 4096ppr incremental encoder accelerating from 0 to 2000rpm at a rate of 10000rpm/s.

Safety Information
Introduction
Installation
Getting started
SM-Register Functionality
Parameters
DPL Programming
Communications
Freeze and Marker
CTSync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

## 5.2 Registration sensor connections

Figure 5-2 SM-Register - Front view



The terminal functions are given in Table 5.3.

**NOTE** Please ensure that the drive is off before adding or removing any modules. Please refer to your installation sheet for more information.

Table 5.3 Module connectors

Terminal	Function	Description
9	0V	0V connection for sensor inputs.
10	DIO	SM-Register Channel 0 sensor input.
11	DI1	SM-Register Channel 1 sensor input.

Table 5.4 Digital input specifications

Terminal 10 / SM-Register Channel 0 Sensor Input	
Terminal 11 / SM-Register Channel 1 Sensor Input	
Type	Positive logic IEC 61131-2.
Maximum Input Voltage	+/- 30V.
Switching Threshold	9.5V +/- 0.3V.
Load	2mA at +15V.

**NOTE** Where sensors with open collector style sensor outputs are used, e.g. photo electric switches, ensure that sufficiently effective pull up / pull down resistors are installed, such that the rising and falling edge times are <250µs (ideally in the order of 1µs), to allow the SM-Register's sensor trim timing scheme to compensate for the sensors' output delay. For most sensors the rising edge time will be in the order of 1µs, however the falling edge time can be much longer depending on the pull down resistor used. This is particularly important in high speed systems.

## 5.3 Functional description

The aim of this section is to provide the user with a practical insight in to the intended use of the SM-Register position capture system, in order to develop an application based upon its features.

### 5.3.1 Sensor setup

In order to get the best performance from the SM-Register position capture, compensation for the given sensors' response time must be applied. The sensor response time is made up from the sum of input circuitry delay, signal processing time, and output circuitry delay. Failure to compensate for this could result in inaccurate results at high speed, e.g:

A sensor with a response time of 40µs, observes a single mark on a drum of circumference 500000µm. When the drum turns at 1m/min the position captured with respect to 1 revolution is 200000µm. At 600m/min however the response time in the sensor causes an offset equal to  $(600\text{m/min} \times 1000000 \times 40\mu\text{s}) / (60 \times 1000000) = 400\mu\text{m}$  or 0.4mm, giving a captured position of 200400µm.

The manufacturers' specification sheet for the chosen sensor should have the response time listed, however if it is not listed an alternative method of determining the response time is listed below:

1. Set up a motor and drive. The motor output shaft must have a pulley or similar disk attached with a single detectable mark.
2. Set the axis turning at a slow speed e.g. 10 rpm.
3. Configure a registration channel to look at the sensor output, using position information only (no turns information), looking for positive pulses.
4. Sample the rising and falling edge positions picked up by the sensor, and filter the result. When the filtered result stabilises, make a note of the position values.
5. Change the axis speed from 10rpm to the maximum speed possible e.g. 3000rpm.
6. Follow step 4, making a note of the stabilised filtered results.
7. Calculate the difference between the slow speed positions captured and the ones at high speed (Slow High Diff). There will be a difference for the rising and falling edges.
8. Calculate the response time from:  
Response time = Slow High Diff \*  $(1 / ((3000\text{rpm}/60) * \text{UPR}))$   
(UPR is Units per revolution).
9. Convert the result to time in nanoseconds, ready to configure the SM-Register system.

Once the response times for the rising edge and falling have been established, they are entered in to the system using the **SetRegistrationTrimTimings** function block. Both RisingEdgeTrim% and FallingEdgeTrim% are in nanoseconds, and when set will reduce the effect of sensor response time to the nearest 139ns interval; 139ns is the smallest internal interval.

Where sensors with open collector style sensor outputs are used, e.g. photo electric switches, ensure that a sufficiently effective pull up / pull down resistor is installed, such that the rising and falling edge times are both <250µs (ideally in the order of 1µs), to allow the SM-Register's sensor trim timing scheme to compensate for the sensors' output delay. For most sensors the rising edge time will be in the order of 1µs, however the falling edge time can be much longer depending on the pull down resistor used. This is particularly important in high speed systems.

When choosing a sensor for high speed and high accuracy, it is important to select a device with minimal jitter since a few microseconds of jitter can result in large position differences e.g:

A sensor with a jitter envelope time of 16µs, observes a single mark on a drum of circumference 500000µm. When the drum turns at 1000m/min the position output may vary by up to  $(1000\text{m/min} \times 1000000 \times 16\mu\text{s}) / (60 \times 1000000) = 267\mu\text{m}$  or 0.267mm.

Safety Information
Introduction
Installation
Getting started
<b>SM-Register Functionality</b>
Parameters
Programming
DP-PL
Communications
Freeze and Marker
CTSync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

### 5.3.2 DPL code

When writing the SM-Register position capture configuration software it is important to select the correct processing task in which to insert the code. It is recommended that if the SM-Register module is only capturing position and calculating a registration position offset then the Background task should be used. However, if there is other code present in the Background task which is slowing down the cycle time e.g. floating point calculations or system sequencing etc., it is recommended that the Clock task is used instead, to make sure the registration code has a higher priority and is serviced at a regular interval.

**NOTE** The Pos tasks are not to be used for SM-Register code, since a SLx.Er error (TR54 - DPL Task Overrun) is much more likely to occur.

When writing software using the SM-Register function blocks, the Status% output from each of the function blocks should be verified by the user. This is because the information on the success / failure of any operation which may be performed using the SM-Register system is only available via the Status% output i.e. the drive will not trip if there is a setup issue.

Incorporating Status% checks in to the registration system code, at the beginning, can help save time in the long run when identifying issues when commissioning the system, in a real application. To do this effectively a separate identifier should be made for each individual Status% output e.g. Status1%, Status2% etc.

For certain operations checking the Status% is mandatory e.g. **GetRegistrationEvent** and **GetRegistrationFrame**, where the Status% output indicates when valid data is available to use.

For phase advance / retard or alignment type systems a small additional calculation is required to calculate the registration position offset. These calculations vary depending on the application, however two basic types are shown below:

- Registration Offset = Required Position - Actual Captured Position  
(This is a position based calculation, and is used in digital lock based applications such as Printing, or simple position profiled applications like edge trimmers).
- Registration Offset = (Current accepted event position - Last Accepted event position) - Product Length.  
(This is a length based calculation, and is used in digital lock based applications such as Printing, but can also be used in more complex CAM based applications like Rotary Knife / Shear or Flying Shear).

### 5.3.3 Choosing a position data source for capture

Each SM-Register capture channel may be directed to capture position data from one of two encoder objects; the Reference **or** the Feedback encoder object.

To configure the SM-Register module to use the Reference or the Feedback object set the PositionDataSource% input for the **ConfigureRegistrationMode** function block, where:

- PositionDataSource% = 0 for the Reference object.
- PositionDataSource% = 1 for the Feedback object.

To configure the Reference or Feedback encoder position data source, use either:

- #90.44 or APCSetReferenceSource(Src%) for the Reference encoder, or
- #90.43 or APCSetFeedbackSource(Src%) for the Feedback encoder.

To select one of the 5 available position data sources set #90.43, #90.44 or Src% from 0 to 4, where:

- Drive D Type = 0.
- Option slot 1 = 1.
- Option slot 2 = 2.
- Option slot 3 = 3.
- User Program = 4.

If the Reference or Feedback encoder object directed to the user program (#90.44 = 4, or #90.43 = 4, or Src% = 4), then a position counter **MUST** be provided by the user via the APCSetReferencePosition or APCSetFeedbackPosition command in the POS0 task.

5.3.4 Choosing position units

Each SM-Register capture channel has an independent units per revolution scaler, allowing the user to set whatever units suit the application, e.g:

If an application roller has a circumference of 500000µm the units per revolution scaler may be set to 500000 so that all reported positions will be in micrometer units. Optionally, the user may set the resolution as a number of turns bits from 0 to 16, where the units per revolution are defined by 2<sup>(32 - Turns Bits)</sup> e.g. if 16 turns bits are set there will be 16 bits of information remaining in the 32 bit signed integer data format, therefore the number of units per revolution will be 2<sup>16</sup> or 65536.

To configure the scaling set the PositionUnits% input for the

**ConfigureRegistrationMode** function block, where:

- PositionUnits% > 0, the number of units per revolution is set.
- PositionUnits% 0 to -16, the number of turns bits is set e.g. -16 = 16 turns bits, and the number of units per revolution = 2<sup>(32 + PositionUnits%)</sup>

**NOTE** If *PositionUnits%* is positive and not a power of 2, e.g. 5000, then turns information may only be used for applications whose total travel is less than (2<sup>32</sup> / PositionUnits%) revolutions. If this total distance is exceeded a discontinuity will be seen in the reported position; for this reason most rotary applications should not use turns information if *PositionUnits%* is positive and not a power of 2. Where PositionUnits% is a positive non power of 2, and turns information is selected (PositionFormat% = 0), the **ConfigureRegistrationMode** function block will return a Status% of 2, indicating that the axis has been configured, but care must be taken that the axis position does not go past 2<sup>31</sup>-1 in the forwards direction, or past -2<sup>31</sup> in the reverse direction.

Scaling the position units in this way enables any position errors / offsets calculated by the SM-Register module, to be passed directly to one of the Control Techniques motion platforms e.g. APC (Advanced Position Controller) or PLCopen, without the need for units conversion, e.g:

- If PositionUnits% = 360, then to pass an offset directly to the APC, set the APC units per revolution scaling using:  
SetUPRNumeratorAndDenominator(Num%,Den%), where Num% = 360, and Den% = 1.
- If PositionUnits% = 360, then to pass an offset directly to the PLCopen, set the PLCopen units per revolution scaling using:  
EnablePLCopenMode(MotionEngineRate%, UPRNumerator%, UPRDenominator%, AbsoluteEncoder%, EncoderCountsPerRev%, ControlMode%), where UPRNumerator% = 360, and UPRDenominator% = 1.

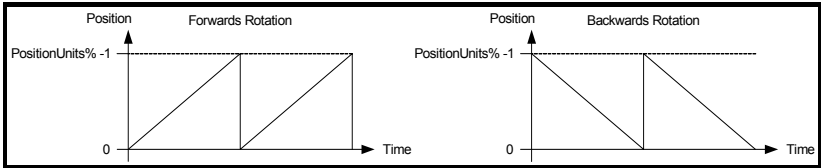
If the unit scaling of the motion platform differs from the SM-Register scaling, then a conversion calculation will be required in the DPL Code before the offset can be used.

### 5.3.5 Choosing the position format

The SM-Register module is able to display results or receive setup information in two basic formats; with turns information and without turns information.

If turns information is not selected (position information only), the range of positions that may be captured will range from 0 to  $\text{PositionUnits}\% - 1$ . The diagram below shows the range of positions that may be captured from the position data source (the axis is turned at a constant speed).

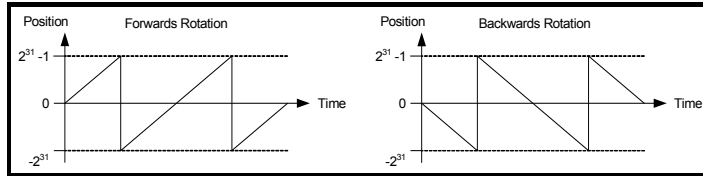
**Figure 5-3 No turns position format**



This type of operation is ideal for rotary applications like printing or rotary knife.

If turns information is selected, the general range of position values that may be captured will range from  $-2^{31}$  to  $2^{31}-1$ . This means that both positive and negative values for the position will be received. The diagram below shows the range of positions that may be captured from the position data source (the axis is turned at a constant speed).

**Figure 5-4 With turns position format**



This type of operation is ideal for linear applications like conveyor handling systems or Flying Shears.

To configure the SM-Register module to use turns information or not, set the `PositionFormat%` input for the **ConfigureRegistrationMode** function block, where:

- `PositionFormat% = 0`, turns and position information is selected
- `PositionFormat% = 1`, position information is selected (no turns information)

**NOTE**

If `PositionUnits%` is positive and not a power of 2, e.g. 5000, then turns information may only be used for applications whose total travel is less than  $(2^{32} / \text{PositionUnits}\%)$  revolutions. If this total distance is exceeded a discontinuity will be seen in the reported position; for this reason most rotary applications should not use turns information if `PositionUnits%` is positive and not a power of 2. Where `PositionUnits%` is a positive non power of 2, and turns information is selected (`PositionFormat% = 0`), the **ConfigureRegistrationMode** function block will return a `Status%` of 2, indicating that the axis has been configured, but care must be taken to ensure that the axis position does not go past  $2^{31}-1$  in the forwards direction, or past  $-2^{31}$  in the reverse direction.

### 5.3.6 Choosing an operating mode

The SM-Register module has seven different capture modes:

- Negative pulses manual (1).
- Positive pulses manual (2).
- Negative pulses automated (3).
- Positive pulses automated (4).
- Negative edges manual (5).
- Positive edges manual (6).
- Positive and Negative edges manual (7).

These capture modes are split in to two types:

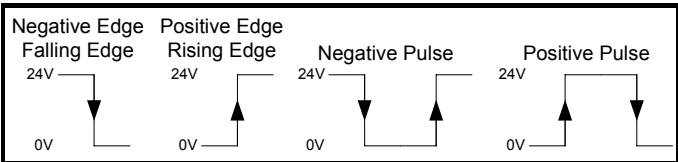
**Manual capture:** The manual capture types are for basic edge or pulse capture. In the manual modes, data is collected from the system using the **GetRegistrationEvent** function block. In a manual pulse capture mode, leading edge position, pulse width, and trailing edge position information is available. In a manual edge capture mode, the edge position, and the distance from the previous pulse seen is available (except the first edge seen, where only the position is available, since no previous edge has happened at that point).

**Automated capture:** The automated capture types are for pulse pattern recognition or frame capture. In the automated modes, data is collected from the system using the **GetRegistrationFrame** function block. The user must define the pattern of pulses or data frame to be captured before enabling capture using the **SetRegistrationStartCondition** function block. A unique starting or trigger condition of up to 8 pulses and gaps may be specified to start capture, with up to 256 pulses in total that may be reported inclusive of the trigger condition pulses e.g. if the trigger condition contains 4 pulses an additional 252 pulses may be reported in the results array. For each pulse reported in the **GetRegistrationFrame** OutputArray%, the user can choose between 4 different data presentation modes which will report various combinations of absolute pulse position, absolute pulse width, pulse distance from the start of the frame, and distance from the start of the previous pulse, by selecting the appropriate **GetRegistrationFrame** Mode%.

**NOTE**

- A negative edge is defined as a 24V to 0V transition.
- A positive edge is defined as a 0V to 24V transition.
- A negative pulse is defined as a 24V to 0V transition, followed by a 0V to 24V transition.
- A positive pulse is defined as a 0V to 24V transition, followed by a 24V to 0V transition.

**Figure 5-5 Pulse and edge definitions**



The manual modes are for general purpose single event capture applications like labeling machines, edge trimmers, or flying shears. It is possible for groups of pulses to be handled using this method, however this would have to be handled in user code. The automated modes are most suited to print based applications like synchronising print rollers, or sheeting printed products where the print register marks are used to define where the sheet cut point is.

Safety Information
Introduction
Installation
Getting started
<b>SM-Register functionality</b>
Parameters
DP-PL Programming
Communications
Freeze and Marker
CT-Sync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

To configure the SM-Register module to use one of the capture types, set the `OperationMode%` input for the **ConfigureRegistrationMode** function block, where:

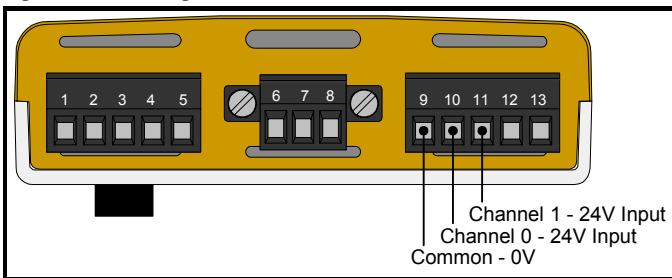
- `OperationMode% = 1`, manual negative pulses are selected.
- `OperationMode% = 2`, manual positive pulses are selected.
- `OperationMode% = 3`, automated negative pulses are selected.
- `OperationMode% = 4`, automated positive pulses are selected.
- `OperationMode% = 5`, manual negative edges are selected.
- `OperationMode% = 6`, manual positive edges are selected.
- `OperationMode% = 7`, manual positive and negative edges are selected.

### 5.3.7 Choosing a capture channel

The SM-Register module has two capture / sensor input channels. Each channel is completely independent of the other, and may be set up entirely differently, or exactly the same, using any of the available capture configurations and options.

The user can connect the 24V output from the registration sensor either directly, or via a standard industrial optical isolator to the following terminals:

**Figure 5-6 SM-Register connection terminals**



- Terminal 9 = Common sensor 0V.
- Terminal 10 = Channel 0, 24V sensor input.
- Terminal 11 = Channel 1, 24V sensor input.

To configure the SM-Register module to use one of the capture channels, set the `ChannelNumber%` input for the **ConfigureRegistrationMode** function block, where:

- `ChannelNumber% = 0`, capture channel 0 will be configured for use.
- `ChannelNumber% = 1`, capture channel 1 will be configured for use.

**NOTE** If both channels are used at the same time then the processing resource used during operation will approximately double, therefore the maximum number of peak and burst edges the system can handle will be reduced.

### 5.3.8 Setting up automated capture

The SM-Register Module has two automated capturing modes; one for capturing patterns or frames of negative pulses, and one for capturing patterns or frames of positive pulses. The **ConfigureRegistrationMode** function block is used to select either of these modes.

Before the configured channel can be enabled to capture event frames, the system needs a description of a unique pulse or pattern of pulses that can be used to trigger the automated capture, and a quantity of pulses to report; this is called a Start Condition. The start condition data is entered in to an integer array (`StartCondition%[n]`), where:

**StartCondition%[0]** = The number of pulses in the trigger, from 1 to 8.



**StartCondition%[1]** = The total number of pulses to report in the **GetRegistrationFrame** **OutputArray%**, from 1 to 256 including the trigger pulses, e.g. if there are 3 pulses in the trigger, the first 3 results in **OutputArray%** will be the positions and widths of the trigger pulses, with a maximum of up to 253 additional pulses reported that occur after the trigger.

**StartCondition%[2]** = The minimum distance from the start of the previous pulse that trigger pulse 1 should be accepted.

**StartCondition%[3]** = The maximum distance from the start of the previous pulse that trigger pulse 1 should be accepted.

**StartCondition%[4]** = The minimum pulse width for trigger pulse 1.

**StartCondition%[5]** = The minimum pulse width for trigger pulse 1.

**NOTE** Normally, for trigger pulse 1, the maximum and minimum distance from the previous pulse will be irrelevant, so 0 should be entered for both minimum and maximum distance to tell the registration system to discount the distance from the previous pulse. However, if a pattern is only to be accepted if it lies a specified distance from any previous mark, then distance from previous pulse may be entered, e.g. minimum of 30mm from previous pulse and maximum of 100mm.  
If this is the case, then on enabling the channel there will have been no previous pulses, so immediately seeing the intended trigger pattern (apart from first pulse distance from previous pulse) will not result in a complete match. The second time the trigger condition is seen, it would result in a complete match and the frame being reported to the user in the **GetRegistrationFrame** **OutputArray%**.

The format of array elements 2 to 5 is repeated for every additional pulse in the trigger, so in general form where n is the pulse number from 1 to 8:

**StartCondition%[4n-2]** = The minimum distance from the start of the previous pulse that trigger pulse n should be accepted.

**StartCondition%[4n-1]** = The maximum distance from the start of the previous pulse that trigger pulse n should be accepted.

**StartCondition%[4n]** = The minimum pulse width for trigger pulse n.

**StartCondition%[4n+1]** = The minimum pulse width for trigger pulse n.

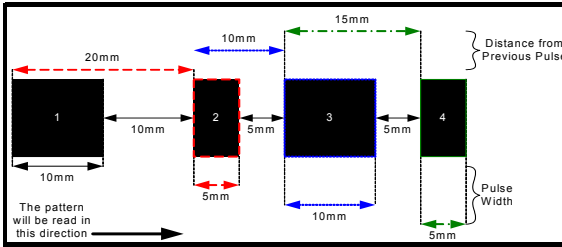
**NOTE** The minimum and maximum distance from the previous pulse is specified in this way to minimise the effect material stretch, when looking for patterns of pulses and gaps e.g. when printing on to plastic film. Other methods of specification like using the distance from the first pulse to specify the gaps, would require progressively wider limits as the end of the pattern is reached.

**NOTE** All widths and distances are positive integer values.

An example of a start condition is shown below, which consists of:

- Four pulses in the trigger.
- Four pulses to be reported in the **GetRegistrationFrame** **OutputArray%**.
- A +/-1mm tolerance for each pulse and distance between pulses.
- The units per revolution have been scaled to give 1 micrometer (0.001mm) units.

**Figure 5-7 StartCondition% example**



StartCondition%[0] = 4	Number of trigger pulses
StartCondition%[1] = 4	Number of reported pulses
StartCondition%[2] = 0 (no previous pulse)	Pulse 1 setup data
StartCondition%[3] = 0 (no previous pulse)	
StartCondition%[4] = 9000 (10mm - 1mm)	
StartCondition%[5] = 11000 (10mm + 1mm)	
StartCondition%[6] = 19000 (20mm - 1mm)	Pulse 2 setup data
StartCondition%[7] = 21000 (20mm + 1mm)	
StartCondition%[8] = 4000 (5mm - 1mm)	
StartCondition%[9] = 6000 (5mm + 1mm)	
StartCondition%[10] = 9000 (10mm - 1mm)	Pulse 3 setup data
StartCondition%[11] = 11000 (10mm + 1mm)	
StartCondition%[12] = 9000 (10mm - 1mm)	
StartCondition%[13] = 11000 (10mm + 1mm)	
StartCondition%[14] = 14000 (15mm - 1mm)	Pulse 4 setup data
StartCondition%[15] = 16000 (15mm + 1mm)	
StartCondition%[16] = 4000 (5mm - 1mm)	
StartCondition%[17] = 6000 (5mm + 1mm)	

To configure the SM-Register module to use the StartCondition% array, use the Status% = **SetRegistrationStartCondition**(StartCondition%,ChannelNumber%) function block. This must be done before enabling capture using **EnableRegistrationMode**.

### Commissioning

When commissioning automated capture, a good method of verifying that the StartCondition% array has been set up correctly, is to:

1. Reduce the number of trigger pulses and reported pulses to 1.
  2. Verify that event frames are being received.
  3. Increase the number of trigger pulses and reported pulses by 1, then repeat step 2.
- Follow this method until the complete trigger is verified.

When choosing a pattern of pulses to trigger capture on, it is essential that this pattern is unique, and that it is not repeated at any point before the total number of reported pulses have been seen. If a trigger pattern is seen before the total number of reported pulses have happened, the pattern capture will be reset using the most recent trigger pattern for the capture. If a new trigger pattern is always seen before the total number of reported pulses is seen, then no event frames will be reported, since the trigger will be continually reset.

### 5.3.9 Enabling and disabling capture

Once the capture has been fully configured capture may be enabled and disabled at will, using `Status% = EnableRegistrationMode(ChannelNumber%)` to turn on capture for the configured channel (each channel must be separately configured), and `Status% = DisableRegistrationMode(ChannelNumber%)` to turn off capture for the configured channel.

**NOTE** If a channel is configured, but disabled it will not use processor resource. Only when the channel is enabled will processor resource be consumed.

For users of the SM-Applications series of modules, that are familiar with how the Freeze's latching position capture functionality operates, the SM-Register **EnableRegistrationMode** and **DisableRegistrationMode** commands operate in a "similar" way to the Freeze capture flag. See the table below:

Freeze Operation	Similar SM-Register Operation
Reset the Freeze flag from 1 to 0.	User calls <b>EnableRegistrationMode</b> , when the SM-Register capture system was previously disabled.
An event happens which latches the Freeze flag.	An event happens, and the user calls <b>DisableRegistrationMode</b> .
When the Freeze flag is latched, and when <b>DisableRegistrationMode</b> is called, no events will be seen.	

When the SM-Register system is disabled and then re-enabled, any previous edge or pulse data in the system will be reset. This may affect certain operations which store the previously reported event, e.g. the registration filter in an edge mode, where edges are filtered by comparing to the previous accepted edge.

### 5.3.10 Collecting captured events

The method used to collect any captured events depends on the type of capture that has been configured; manual or automated capture.

- Manual capture uses `GetRegistrationEvent` to collect event data.
- Automated capture uses `GetRegistrationFrame` to collect patterns of event data.

#### Manual events

There are 5 types of manual event that the user can configure the SM-Register position capture system to look for:

- Negative pulses.
- Positive pulses.
- Negative edges.
- Positive edges.
- Negative and positive edges.

When data is collected from the configured and enabled channel using the **GetRegistrationEvent** function block, the user will be presented with five pieces of data:

**Status%** - This will change from 0 to 1 when a new event has been captured, and is intended to be used to tell the user code when to process the other outputs. When **Status%** is set to 0, the **EventDescriptor%**, **LeadingPosition%**, **Distance%** and **TrailingPosition%** outputs will be set to 0; this means that the user processing of the data must happen on the cycle in which the result is read.

If more than 256 events are detected between successive calls of the **GetRegistrationEvent** function block, or the rate at which pulses are seen is greater than the rate at which they are read out, the internal 256 place buffer will wrap over, and begin to overwrite the first results; in this event a Status% of 2 will be reported instead of 1, indicating that data has been over-written.

**EventDescriptor%** - This is an identifier for the type of data received, where:

- 1 = Negative Edge.
- 2 = Positive Edge.
- 3 = Negative Pulse.
- 4 = Positive Pulse.

This is primarily intended to be used when negative and positive edge capture mode is selected, so that the user code can tell whether a negative edge or a positive edge has been received.

**LeadingPosition%** - This indicates the leading event position. For a negative pulse this will be the falling edge position, for a positive pulse this will be the rising edge position, and for an edge capture it will show the position of the captured edge.

**Distance%** - This indicates the width of a detected pulse, if a pulse capture mode is selected. If an edge mode is selected, it indicates the distance from the previous edge; for the very first edge after enabling capture this will contain 0, since there are no previous edges to compare with.

**TrailingPosition%** - This indicates trailing event position. For a negative pulse it will show the rising edge position, for a positive pulse it will show the falling edge position, and for an edge capture it will be clamped at 0.

#### Automated events

There are 2 types of automated event that the user can configure the SM-Register position capture system to look for:

- Negative pulses (single pulse or pattern of pulses).
- Positive pulses (single pulse or pattern of pulses).

When data is collected from the configured and enabled channel using the **GetRegistrationFrame** function block, the user must:

Choose a reporting format - The user can choose from 4 different result reporting formats by setting the Mode% input to the **GetRegistrationFrame** function block, which include:

##### 1 or **ABSOLUTE\_MODE**.

The pulse data described using their absolute pulse width and position. The frame will be reported as pulses consisting of:

[2n-1] Absolute pulse position.

[2n] Absolute pulse width.

(n is the pulse number, starting from pulse 1).

##### 2 or **FRAME\_RELATIVE\_MODE**.

The pulse positions will be reported relative to the start of the frame, with the exception of the first pulse which will be reported as an absolute position. The frame will be reported as pulses consisting of:

[2n-1] Pulse distance from start of frame.

[2n] Absolute pulse width.

(n is the pulse number, starting from pulse 1).

**3 or PULSE\_RELATIVE\_MODE.**

The pulse positions will be reported relative to the previous pulse, with the exception of the first pulse which will be reported as an absolute position. The frame will be reported as pulses consisting of:

[2n-1] Pulse distance from start of previous pulse.

[2n] Absolute pulse width.

(n is the pulse number, starting from pulse 1).

**4 or HYBRID\_MODE.**

The pulse positions will be reported in all of the above modes for greatest flexibility. The pulse positions within the first pulse will be reported in absolute mode. The frame will be reported as pulses consisting of:

[4n-3] Absolute pulse position.

[4n-2] Pulse distance from start of frame.

[4n-1] Pulse distance from start of previous pulse.

[4n] Absolute pulse width.

(n is the pulse number, starting from pulse 1).

Dimension an OutputArray% - The captured frame data will be placed in to this array; the DIM command is used dimension an array, e.g. DIM OutputArray%(30) dimensions a 30 element array called OutputArray%. The size of this array depends upon the number of pulses to be reported and the reporting mode chosen, e.g. if there are eight pulses to be reported and reporting mode four is selected, the OutputArray% must have 32 elements.

The user then calls the **GetRegistrationFrame** function block in DPL, and will then be presented with the following data:

**Status%** - This will change from 0 to 1 when a new data frame has been captured, and is intended to be used to tell the user code when to process the handle the new data in the OutputArray%. When Status% is set to 0, the data in the OutputArray% will remain unchanged until the next data frame is captured.

If the OutputArray% has not been made big enough for the data frame defined by the **SetRegistrationStartCondition** function block, a Status% of 2 will be reported, indicating that as many elements as possible have been filled with data, however, a complete set could not be written; if this happens NumberOfEvents% will show the number of completed sets of pulse data.

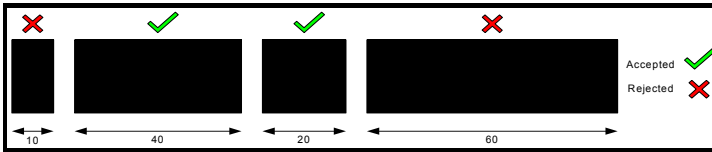
**NumberOfEvents%** - This displays the number of complete pulse results that will be seen in the OutputArray%. Under normal circumstances NumberOfEvents% = StartCondition%[1] used by the **SetRegistrationStartCondition** function block.

**5.3.11 Filtering event data**

The SM-Register position capture system has a comprehensive filtering scheme that automatically accepts or rejects events (pulses and edges), without the need for extensive user code. The MinimumDistance% and MaximumDistance% inputs for the **ConfigureRegistrationFilter** function block allow the user to setup the thresholds of the filter, however, the operation depends upon whether pulses or edges are being detected:

**Pulse capture** - in a pulse capture mode, the MinimumDistance% and MaximumDistance% define the minimum and maximum pulse width that will be reported, e.g. MinimumDistance% = 20 and MaximumDistance% = 50, only pulses with a width between these two limits will be reported to the user.

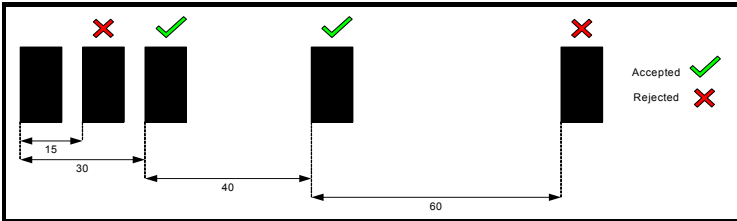
**Figure 5-8 Pulse filtering**



Additionally the user may disable one or other of the limits by setting a value of 0 in either MinimumDistance% or MaximumDistance%; if MinimumDistance% = 0, then pulse widths  $\leq$  MaximumDistance% are reported, or if MaximumDistance% = 0, then pulse widths  $\geq$  MinimumDistance% will be reported.

**Edge capture** - in an edge capture mode, the MinimumDistance% and MaximumDistance% define the minimum and maximum distance from the previously reported pulse that a new pulse will be reported, e.g. MinimumDistance% = 20 and MaximumDistance% = 50, only edges seen between these distances after the previously reported edge will be made available the user.

**Figure 5-9 Edge filtering**



Additionally the user may disable one or other of the limits by setting a value of 0 in either MinimumDistance% or MaximumDistance%; if MinimumDistance% = 0, then pulse widths  $\leq$  MaximumDistance% are reported, or if MaximumDistance% = 0, then pulse widths  $\geq$  MinimumDistance% will be reported.

**NOTE**

When the filter is first enabled, the initial pulse seen is always reported to the user, as there is no previous edge to do a distance comparison with.

Once MinimumDistance% and MaximumDistance% have been configured for the appropriate ChannelNumber%, the user may then enable or disable the filter by using the **EnableRegistrationFilter** or **DisableRegistrationFilter** function blocks. This may be done with capturing enabled or disabled.

The user can re-configure the minimum and maximum filter distances on the fly while the filter is enabled, however, the change will not take affect till the beginning of the next position update cycle.

### 5.3.12 Windowing event data

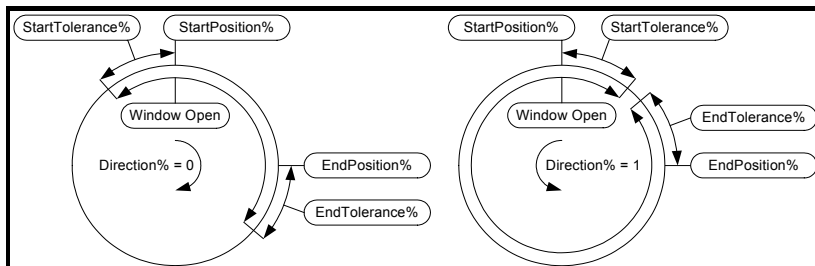
The SM-Register position capture system has a comprehensive windowing scheme that automatically accepts complete events (pulses and edges), that are seen inside a configurable position window (StartPosition% to EndPosition%), and rejects all events outside of the window without the need for extensive user code. This differs from the registration filter in two main ways:

- Filtering is applied by accepting any events between two positions, rather than the dimensions of the event or the distance from the previous event.
- Windowing saves processor resource since capture is completely disabled outside of the window region. Filtering works by assessing each event, which means resource is taken for every event seen, whether it is filtered out or reported to the user.

**NOTE** For some high speed applications with a large throughput of events, the window may be required to remove unwanted events, so that the SM-Register processor resource will not be completely used, preventing an SLx.Er error (TR54 - DPL Task Overrun).

The **ConfigureRegistrationWindow** function block is used to setup the windowing scheme, where the user must decide upon a StartPosition%, StartTolerance%, EndPosition%, EndTolerance%, and a Direction% in which to specify the window; the diagram below illustrates the relationship between these inputs:

**Figure 5-10 Registration window setup**



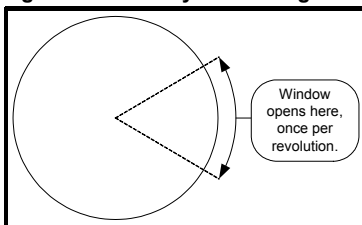
In Figure 5-10 it can be seen that for the forwards direction (Direction% = 0), that the window open area is defined by  $(\text{EndPosition}\% + \text{EndTolerance}\%) - (\text{StartPosition}\% - \text{StartTolerance}\%)$ , and that for the reverse direction (Direction% = 1), the window open area is defined by  $(\text{StartPosition}\% + \text{StartTolerance}\%) - (\text{EndPosition}\% - \text{EndTolerance}\%)$ . The inputs are defined in this way to minimise the number of calculations the user must do; this is particularly important when turns information is not used, as deriving positions or distances can become complicated.

In situations where a calculation must be done e.g. when working out the EndPosition% in an automated capture mode when no turns information is selected, the **CalculateRegistrationPosition** and **CalculateRegistrationDistance** function blocks may be used to handle the calculation for the user.

**NOTE** Direction% is only used when specifying the window; it does not define the direction of rotation for the final application.

For an application that uses rotary windowing e.g. Printing, once the window has been set, it will not need altering since these applications are most suited to the no turns PositionFormat% i.e. the area in which the window will fall never changes.

**Figure 5-11 Rotary windowing**



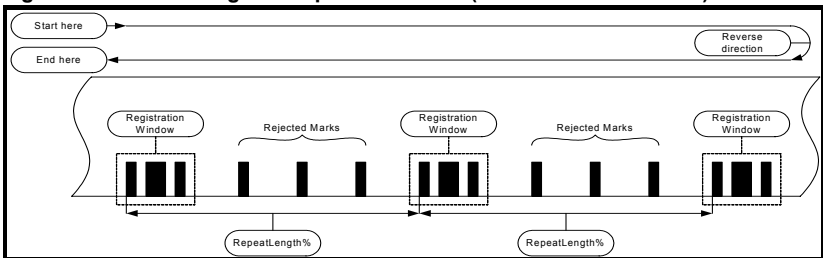
**NOTE** The window must be long enough to be open for at least 3 position update cycles at any given speed, e.g. an application runs at 600m/min, and a position update cycle of 1ms, giving a minimum window length of:

$$\text{Min Length} = (600 / (60 * 1000)) * 3 = 0.03\text{m} = 30\text{mm}.$$

For a linear application that uses windowing e.g. Flying Shear, the window must be moved along with the process, as they typically use turns information i.e. if the window were specified once, with turns information selected, it would only open once in the complete range of positions from  $-2^{31}$  to  $2^{31}-1$ .

To move the windowed area, the user may employ the RepeatDistance% input for the **ConfigureRegistrationWindow** function block, which moves the window automatically by RepeatDistance% units in the current direction of travel, ready to pick up the next event i.e. if the direction of rotation is reversed the window will still be placed correctly.

**Figure 5-12 Windowing with repeatdistance% (with turns information)**



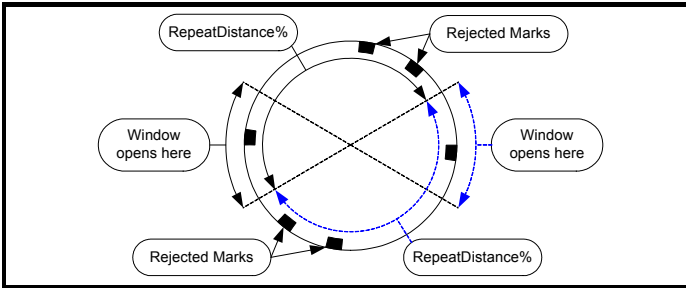
When turns information is selected, RepeatDistance% must be  $\geq$  total window length to move the window automatically. To prevent the repeat distance from moving the window, set RepeatDistance% to 0.

**NOTE** Where turns information is used, and if PositionUnits% is a positive non power of 2 value, care must be taken that the total travel does not exceed  $-2^{31}$  or  $2^{31}-1$ , otherwise a discontinuity may be seen. Where restricting the travel length is not practical, it is recommend that PositionUnits% is set to a negative value, which puts the windowing system into turns bits scaled mode which will wrap over the  $2^{31}-1$  to  $-2^{31}$  rollover point without a discontinuity in the reported values. This may require some user scaling to convert from  $2^x$  units per revolution, to the application measurement units e.g. mm.

A RepeatLength% may also be set for applications where no turns information is specified. The RepeatDistance% must be  $\geq$  total window length and  $\leq$  PositionUnits%. This may be useful in situations where there is more than one event per revolution, and several marks to be disregarded inbetween. See Figure 5-13.

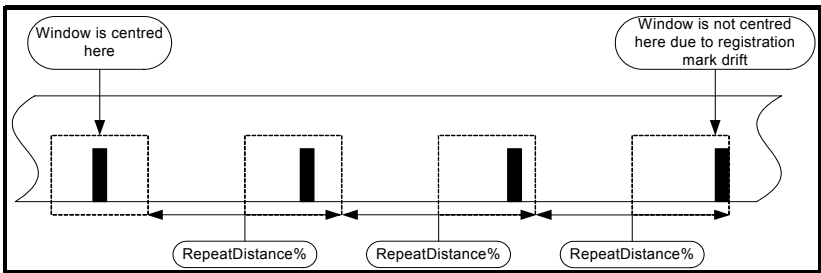


**Figure 5-13 Windowing with repeatdistance% (no turns information)**



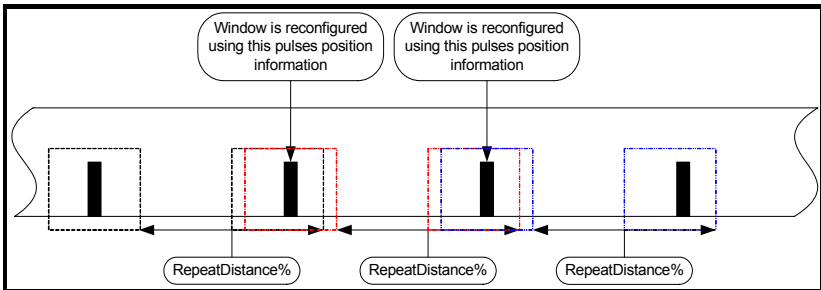
In most linear applications there will be small variations in the distance between windowed events, caused by slight changes in printed position, stretch in the material or even roller slip, that will cause the window to go out of synchronisation with the marks on the product; this is called registration mark drift. See Figure 5-14.

**Figure 5-14 Registration mark drift**



To solve this problem and keep the window aligned, the conventional method used on other systems is to re-configure the window one repeat length on from the beginning of the most recent received event.

**Figure 5-15 Correction for registration mark drift**



This would normally require the user to calculate where the window will next fall based on the current events position data, however, the SM-Register window may be directly reconfigured around the current received event, and still achieve the same effect as if it were configured one repeat length ahead, saving the user from writing additional code. The following method should be used in order to move the window:

1. Receive data on an event in the current window.
2. Set StartPosition% to the current event's leading position.

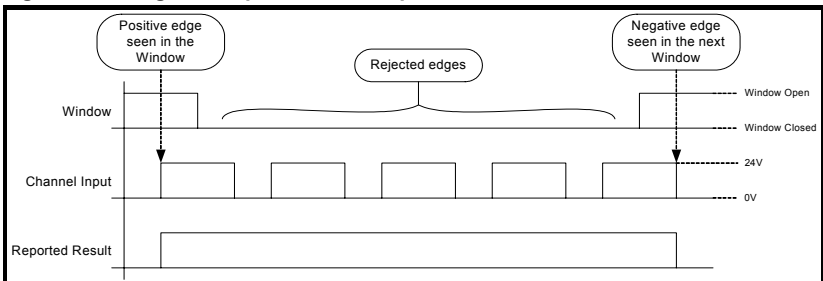
3. Set the EndPosition% to StartPosition% + event width. The **CalculateRegistrationPosition** function block may be used to perform the calculation.
  4. It is assumed that StartTolerance%, EndTolerance%, Direction%, RepeatDistance% and ChannelNumber% will not change.
  5. Re-configure the window using the ConfigureRegistrationWindow function block.
- This process is repeated every time the window opens and an event is reported to the user. If an event is not seen within the window it will continue to advance using the RepeatDistance% alone.
- The user may also calculate and implement a window in advance if preferred.

In situations where a single edge is seen in a window, with another single edge seen at the beginning of the next window, e.g:

- A positive edge in window 1, followed by negative edge in window 2, in a positive pulse mode.
- A negative edge in window 1, followed by a positive edge in window 2, in a negative pulse mode.

The pulse width reported by the user will be the distance between these two edges, even though the window has closed then opened in between. See Figure 5-16.

**Figure 5-16 Edges in separate window periods**



This functionality may be useful in situations where resources are critical, and where the overall width of a group of pulses is sufficient to create a registration offset, rather than receiving each individual pulse / edge.

### 5.3.13 Monitoring functions blocks for SM-Register

The SM-Register has several monitoring function blocks which may be used to get the current settings in use by any of the capture functions; these functions are listed below

#### **GetRegistrationStatus**

This provides the user with the current capture configuration status (Configured%), whether the channel is enabled for capture (Enabled%), what the selected channels position data source is (PositionDataSource%), what the current position units setting is (PositionUnits%), what the current position format is (PositionFormat%), and what the current operation mode is (OperationMode%).

#### **GetRegistrationFilter**

This provides the user with the current filter configuration status (Configured%), whether the registration filter is enabled (Enabled%), the minimum filter distance (Minimum%), and the maximum filter distance (Maximum%).

**GetRegistrationWindow**

This provides the user with the current window configuration status (Configured%), whether the registration window is enabled (Enabled%), whether the selected channel position is in the open area of the window (InWindow%), the minimum window position for the configured channel (Minimum%), the maximum window position for the configured channel (Maximum%), the current configured window length (WindowLength%), the configured repeat distance (RepeatDistance%), and the repeat distance actually in use by the system (RepeatDistanceInUse); this differs from RepeatDistance% in that the user may have configured a bad repeat distance which the system will reject in favour of the previously accepted repeat distance.

**GetRegistrationTrimTimings**

This provides the user with the current rising edge trim time in use (RisingEdgeTrim%), and the falling edge trim time in use (FallingEdgeTrim%). These times may differ from the value set by the user since the value used will be rounded to the nearest 139ns.

**GetRegistrationStartCondition**

This provides the user with the current configured StartCondition% array values in use by the system, provided the StartCondition% has been configured. To use this function the user must dimension a suitably large array to output the results to.

**NOTE** When using the monitoring function blocks it is important to note that the outputs from the function blocks show the values in use, rather than the latest value set by the user. It is possible that if the SM-Register capture system function blocks are running in the background task that several cycles of the background may happen between position update cycles, which for the majority of the SM-Register capture setting is when they will be applied, i.e. the user may set a value, then read back the value straight away, and find that the new setting has not taken effect yet.

An additional monitoring function is available to allow the user to sample the current encoder source Position%, for the configured channel, to the nearest position update cycle, called **GetRegistrationPosition**. This may be useful when diagnosing windowing setup issues, or general system sequencing.

**5.3.14 Position and distance calculation**

The SM-Register capture system has two calculation utility function blocks to aid users when developing a complete registration system, one for position and one for distance:

**CalculateRegistrationPosition**

This function block allows the user to calculate a new Position% from a StartPosition%, Distance% and a Direction% in which to apply the Distance% from the StartPosition%. If Direction% = 0 (forwards), the Distance% will be added to StartPosition%, however, if Direction% = 1 (backwards), the Distance% will be subtracted from StartPosition%.

## CalculateRegistrationDistance

This function block allows the user to calculate a new Distance% from a StartPosition% and an EndPosition%.



The **CalculateRegistrationDistance** function block has the limitation that if the positions do not include turns information, the distance calculated will always be less than one revolution. This might seem obvious but if an event occurred every third revolution on a system in which turns information is not being reported, and those positions were fed into this function the distance would NOT be equal to 3 revolutions - it would be approximately zero as the position within the revolution for each pulse would be approximately identical.

This is the one case where a distance will not exceed one revolution, but that is a consequence of using positions which do not include turns information.

These functions are particularly useful when the PositionFormat% is set to no turns information (position information only), since calculating distance or position in this format can be difficult when handling the position rollover point in the calculations.

### 5.3.15 Corrective action for TR54, or status% = 2 when collecting data

Where an SM-Register module trips the drive on a SLx.Er due to a TR54 (DPL task overrun), and that this can be directly attributed to a high volume of events using up the processor resource, this trip may be prevented using any of the following:

- Reduce the amount of code within the in the Pos 0 or Pos 1 tasks, if they are used, to allow more resources for handling events.
- Remove any motion control code (PLCopen, APC etc.) from the SM-Register module to an additional SM-Applications series module, to reduce processor loading.
- Reduce the event rate by slowing down the process, e.g. reducing the line speed in a printing application.
- Introduce the SM-Register windowing function to prevent the system from seeing unwanted pulses, leaving more processing resource for the ones which are required.
- Reduce the data rate of any communications using CT Net, as a high data rate uses more processing resource.

If the Status% output from either the **GetRegistrationEvent** or **GetRegistrationFrame** function blocks returns a value of 2, the user may do one of the following to help correct the problem:

- Increase the size of the OutputArray% when using the **GetRegistrationFrame** function block.
- Increase the rate at which the results are read in the user code. Provided the **GetRegistrationEvent** or **GetRegistrationFrame** function blocks are called in the Clock task, this may be done by using a faster Clock task scheduling rate by setting #81.11 (x.11) to a lower value. Another method may be to move the **GetRegistrationEvent** or **GetRegistrationFrame** function block calls to the Clock task, in situations where the Background task is heavily loaded with other system code slowing down the Background task cycle time; this works by giving the SM-Register function block calls a higher priority, since the Clock task has a higher priority than the Background task.

#### NOTE

Provided only SM-Register specific commands and associated control code is used when writing the position capture software, the Background task has the highest data collection rates achievable.

## 5.4 Registration function blocks

### 5.4.1 ConfigureRegistrationMode

This function block configures the specified channel for position capture, and clears all internal data, such that when capturing is next successfully enabled no old captured positions will be seen by the user. For this reason the registration mode for a given channel cannot be configured if it is currently enabled as data would be lost. Each channel can have its own encoder source, position units, capture operating mode (pulses / edges). The channel will not start capturing until it is enabled.

The SM-Register capturing system has two basic types of operating mode:

- Manual capture, these capture types are for basic edge or pulse capture.
- Automated capture, these capture types are for pulse pattern recognition / frame capture.

The manual operating modes use “GetRegistrationEvent” to retrieve event data. Automated modes use “GetRegistrationFrame” to retrieve event data.

Status% = ConfigureRegistrationMode(PositionDataSource%, PositionUnits%, PositionFormat%, OperationMode%, ChannelNumber%)

#### Arguments

PositionDataSource%

- 0: This channel should use the reference encoder for position capture.  
1: This channel should use the feedback encoder for position capture.

The feedback or reference encoder location must be configured to a valid position data source, using either #90.44 or APCSetReferenceSource(Src%) for the reference encoder, or #90.43 or APCSetFeedbackSource(Src%) for the feedback encoder. To select one of the 5 available position data sources set #90.43, or #90.44 or Src% from 0 to 4, where the Drive D Type = 0, Option slot 1 = 1, Option slot 2 = 2, Option slot 3 = 3, User Program = 4.

If the PositionDataSource% is the user program (#90.44 = 4, or #90.43 = 4, or Src% = 4) then the position MUST be provided by the user via the APCSetReferencePosition or APCSetFeedbackPosition command in the POS0 task.

PositionUnits%

Positive non-zero values: The number of user units in a revolution, that will be reported in the results, and is used when setting up the registration filter or window. The reported positions are rounded to the nearest whole user position unit. This is referred to as “Full Scale Mode”.

0 to -16: The number of units per revolution that will be reported in the results is defined by  $2^{(32 + \text{PositionUnits}\%)}$  e.g. if PositionUnits% = -16 then the number of units per revolution is given by  $2^{(32 + (-16))} = 65536$  units per revolution. This is referred to as “Turns Bits Scaled Mode”.

The reported positions can include turns information or not, and is defined by the *PositionUnits%* input. If turns bits are specified, it is intended that this setting should be the same as the number of turns bits selected in the module which is controlling the registration motion. This enables any position errors / offsets calculated by the SM-Register module to be passed directly to one of the Control Techniques motion platforms e.g. APC (Advanced Position Controller) or PLCopen, without the need for units conversion, e.g:

- If  $\text{PositionUnits\%} = 360$ , then to pass an offset directly to the APC, set the APC units per revolution scaling using `SetUPRNumeratorAndDenominator(Num%,Den%)`, where  $\text{Num\%} = 360$ , and  $\text{Den\%} = 1$ .
- If  $\text{PositionUnits\%} = 360$ , then to pass an offset directly to the PLCopen, set the PLCopen units per revolution scaling using `EnablePLCopenMode(MotionEngineRate%, UPRNumerator%, UPRDenominator%, AbsoluteEncoder%, EncoderCountsPerRev%, ControlMode%)`, where  $\text{UPRNumerator\%} = 360$ , and  $\text{UPRDenominator\%} = 1$ .

If the unit scaling of the motion platform differs from the SM-Register scaling, then a conversion calculation will be required.

#### PositionFormat%

0: Turns and position information is reported when reading positions. Positions are always in the range  $-2^{31}$  to  $2^{31}-1$ .

1: Turns information is omitted when reading positions. However, distances may be greater than one turn, and may exceed the range of positions given below.

Positions are always in the range:

- If  $\text{PositionUnits\%} > 0$ : 0 to  $(\text{PositionUnits\%} - 1)$ .
- If  $\text{PositionUnits\%} \leq 0$ :  $-2^{(32 + \text{PositionUnits\%})}$  to  $2^{(32 + \text{PositionUnits\%})}-1$ .

#### NOTE

If  $\text{PositionUnits\%}$  is positive and not a power of 2 ("Full Scale Mode" positioning), e.g. 5000, then turns information may only be used for applications whose total travel is less than  $(2^{32} / \text{PositionUnits\%})$  revolutions. If this total distance is exceeded a discontinuity will be seen in the reported position. For most rotary applications turns information must NOT be used in "Full Scale Mode" position. Where  $\text{PositionUnits\%}$  is a positive non power of 2, and turns information is selected, the `ConfigureRegistrationMode` function block will return a  $\text{Status\%}$  of 2, indicating that the axis has been configured, but care must be taken that the axis position does not go past  $2^{31}-1$  in the forwards direction, or past  $-2^{31}$  in the reverse direction.

#### OperationMode%

- 1: This channel will capture negative pulses in manual mode.
- 2: This channel will capture positive pulses in manual mode.
- 3: This channel will capture negative pulses in automated mode.
- 4: This channel will capture positive pulses in automated mode.
- 5: This channel will capture negative edges in manual mode.
- 6: This channel will capture positive edges in manual mode.
- 7: This channel will capture both positive and negative edges in manual mode.

#### ChannelNumber%

- 0: Capture Channel 0, or SM-Register digital input [0].
- 1: Capture Channel 1, or SM-Register digital input [1].

#### Return values

$\text{Status\%}$

See Table 5.5 *Status% Output Table* on page 65.

### 5.4.2 EnableRegistrationMode

This function block enables position capture mode, and clears all internal data, such that when capturing is successfully enabled no old captured positions will be seen by the user. Position capture mode can only be enabled if the channel has been successfully configured in a valid mode. If the specified channel is already enabled when EnableRegistrationMode is called this function block will report a Status% of -15, indicating that it is already enabled.

```
Status% = EnableRegistrationMode(ChannelNumber%)
```

**Arguments**

- ChannelNumber%
- 0: Capture Channel 0, or SM-Register digital input [0].
  - 1: Capture Channel 1, or SM-Register digital input [1].

**Return values**

Status%  
See Table 5.5 *Status% Output Table* on page 65.

### 5.4.3 DisableRegistrationMode

This function block disables position capture mode. If the specified channel is already disabled when DisableRegistrationMode is called, this function block will report a Status% of -19, indicating that it is already disabled.

```
Status% = DisableRegistrationMode(ChannelNumber%)
```

**Arguments**

- ChannelNumber%
- 0: Capture Channel 0, or SM-Register Digin[0].
  - 1: Capture Channel 1, or SM-Register Digin[1].

**Return values**

Status%  
See Table 5.5 *Status% Output Table* on page 65.

### 5.4.4 ConfigureRegistrationFilter

**Pulse Modes:** This function block allows the user to specify the minimum and maximum pulse widths that are accepted on the specified channel. Pulse widths are specified in distance units.

**Edge Modes:** This function block allows the user to specify the minimum and maximum distance between edges, outside of which pulses will be ignored. Distances are specified in distance units.

**All Modes:** The filter may be configured with the channel either enabled or disabled. If this function block is used to reconfigure the filter while the filter is enabled the new filter settings will take effect on the next position update cycle, otherwise the new settings will take effect immediately. Changes also take effect immediately if they have been made from the Initial task, as the registration engine is not running when the Initial task is running.

```
Status% = ConfigureRegistrationFilter(MinDistance%, MaxDistance%, ChannelNumber%)
```

Safety Information	Introduction	Installation	Gating started	SM-Register Functionality	Parameters	DPL	Communications	Freeze and Marker	CTSync	Inter-option synchronisation	Diagnostics	Quick Reference	Index
--------------------	--------------	--------------	----------------	---------------------------	------------	-----	----------------	-------------------	--------	------------------------------	-------------	-----------------	-------

## Arguments

MinDistance%

**Pulse Modes:** The pulse width in units as defined by *PositionUnits%* .

**Edge Modes:** The distance between acceptable edges as defined by *PositionUnits%* .

If this value is zero there is no minimum distance applied.

MaxDistance%

**Pulse Modes:** The pulse width in units as defined in *PositionUnits%* .

**Edge Modes:** The distance between acceptable edges as defined in Position Units.

If this value is zero no maximum distance filter will be applied, but care should be taken that the longest distance is no greater than  $2^{31}$  position units, as an incorrect length would appear to be reported.

ChannelNumber%

0: Capture Channel 0, or SM-Register digital input [0].

1: Capture Channel 1, or SM-Register digital input [1].

## Return values

Status%

See Table 5.5 *Status% Output Table* on page 65.

### 5.4.5 EnableRegistrationFilter

This function block enables the registration filter, if configured, for the specified channel. If the specified channel's filter is already enabled when EnableRegistrationFilter is called this function block will report a Status% of -15, indicating that it is already enabled.

```
Status% = EnableRegistrationFilter(ChannelNumber%)
```

## Arguments

ChannelNumber%

0: Capture Channel 0, or SM-Register digital input [0].

1: Capture Channel 1, or SM-Register digital input [1].

## Return values

Status%

See Table 5.5 *Status% Output Table* on page 65.

### 5.4.6 DisableRegistrationFilter

This function block disables the registration filter for the specified channel, if currently enabled, without modifying the filter minimum and maximum distances. If the specified channel's filter is already disabled when DisableRegistrationFilter is called, this function block will report a Status% of -19, indicating that it is already disabled.

```
Status% = DisableRegistrationFilter(ChannelNumber%)
```

## Arguments

ChannelNumber%

0: Capture Channel 0, or SM-Register digital input [0].

1: Capture Channel 1, or SM-Register digital input [1].

## Return values

Status%

See Table 5.5 *Status% Output Table* on page 65.

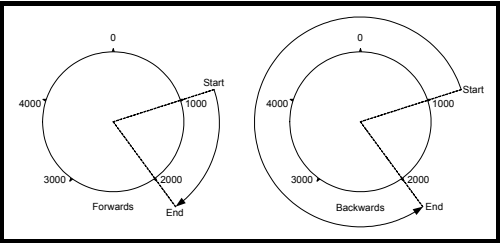


### 5.4.7 ConfigureRegistrationWindow

This function block allows the user to specify either a static or moving registration window, on the specified channel. Inside the registration window capturing is enabled, and outside capturing is disabled rejecting unwanted pulses or edges, and saving processing resource.

Pulses and Edges will only be captured if they are between (StartPosition% - StartTolerance%) and (EndPosition% + EndTolerance%) for a window specified in the forward direction or between (StartPosition% + StartTolerance%) and (EndPosition% - EndTolerance%) for a window specified in the backward direction, when a window is enabled. The Direction% input (Forwards or Backwards) does not specify the direction of travel for the final application, but instead is a means of clearly defining the area the user wants to window, e.g:

If 5000 units per revolution are defined, with no turns information or tolerance, a window start position of 1000 and a window end position of 2000, with the forward direction selected the window area will be 1000 units wide, however with the backwards direction selected it will be 4000 units wide.



The window limits have been specified this way to minimise the number of external calculations the user has to perform when setting up the window, particularly when handling position without turns information.

**NOTE** If turns information is not being used in reported positions, the user must ensure that the window values passed via StartPosition% and EndPosition% do not exceed the valid range of positions. (i.e. never less than zero and never more than the maximum representable position as defined by *PositionUnits%* -1.

This function block may be used to reconfigure the window while the window is still enabled, however the new window settings will only take effect on the next position update cycle, otherwise the changes will take effect immediately. Changes also take effect immediately if they have been made from the Initial task, as the registration engine is not running when the Initial task is running.

Due to the mechanism that the SM-Register module uses to enable and disable edge capture in order to save resource, the window length must be greater than or equal to 3 position update cycles' worth of distance at the present speed, e.g:

For a system using 16384 units per revolution, with a rotary axis of circumference 1 metre, at an angular speed of 600rpm (600m/min, or 163840 units per second) and motion engine period of 1ms, the user must not set a window shorter than 3ms, or 492 position units, which equates to approximately 30mm. At lower speeds the minimum window distance will be proportionally smaller, e.g. 300m/min defines minimum window length requirement of 15mm.

It is not permissible to have a window length which is greater than a non-zero repeat distance.

Safety Information
Introduction
Installation
Getting started
SM-Register Functionality
Parameters
DP-PL Programming
Communications
Freeze and Marker
CTSync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

### RepeatDistance% Operation

A non-zero RepeatDistance% will only take effect when the window length is less than or equal to the repeat distance, AND either:

- Turns information is included in reported positions, OR
- Turns information is NOT included in reported positions but the window repeat distance is less than or equal to the axis length.

When a non-zero value of RepeatDistance% is specified, the system configures two "virtual" windows, one either side of the specified window, at a distance of RepeatDistance% from the basic window and of the same dimensions as the basic window.

When the axis enters either virtual window the basic window is reconfigured to take the position range of the virtual window entered, and the virtual window positions are recalculated. This architecture allows the user to configure the window about any event (Pulse / Edge / Frame) that has been received in the current task cycle, saving the user from having to calculate the next expected window's dimension.

**NOTE** The user does not have to manually add the repeat distance to the observed event position in order to recalculate the new window dimensions.

```
Status% = ConfigureRegistrationWindow(StartPosition%, StartTolerance%, EndPosition%, EndTolerance%, Direction%, RepeatDistance%, ChannelNumber%)
```

### Arguments

#### StartPosition%

The basic position in units as defined in *PositionUnits%* at which positions will start being captured.

#### StartTolerance%

The tolerance to allow (widens the window) at the start of the window.

#### EndPosition%

The basic position in units as defined in *PositionUnits%* at which positions will stop being captured.

#### EndTolerance%

The tolerance to allow (widens the window) at the end of the window.

#### Direction%

The direction of travel for which the window start and end positions have been specified.

0: The positions are described for the FORWARD direction.

1: The positions are described for the BACKWARD direction.

#### RepeatDistance%

The expected repeat distance which, if possible, will be used to move the window automatically. A RepeatDistance% of 0 disables automatic window advance / retard.

#### ChannelNumber%

0: Capture Channel 0, or SM-Register digital input [0].

1: Capture Channel 1, or SM-Register digital input [1].

### Return values

#### Status%

See Table 5.5 *Status% Output Table* on page 65.

5.4.8 EnableRegistrationWindow

This function block causes the window for the specified channel to take effect. Until this function block is successfully called, the window will not have any effect on the number of pulses / edges seen or the processing resource, even if the window has been configured using ConfigureRegistrationWindow.

If the specified channel's window is already enabled when EnableRegistrationWindow is called this function block will report a Status% of -15, indicating that it is already enabled.

```
Status% = EnableRegistrationWindow(ChannelNumber%)
```

Arguments

- ChannelNumber%
- 0: Capture Channel 0, or SM-Register digital input [0].
  - 1: Capture Channel 1, or SM-Register digital input [1].

Return values

Status%  
See Table 5.5 *Status% Output Table* on page 65.

5.4.9 DisableRegistrationWindow

This function block causes the window for the specified channel to have no effect on the captured positions. When the registration window is disabled all edges are “seen”.

If the specified channel's window is already disabled when DisableRegistrationFilter is called, this function block will report a Status% of -19, indicating that it is already disabled.

```
Status% = DisableRegistrationWindow(ChannelNumber%)
```

Arguments

- ChannelNumber%
- 0: Capture Channel 0, or SM-Register digital input [0].
  - 1: Capture Channel 1, or SM-Register digital input [1].

Return values

Status%  
See Table 5.5 *Status% Output Table* on page 65.

5.4.10 GetRegistrationEvent

This function block is used to check for an available event, in the specified capture channel, when the channel is operating in a manual mode e.g. *OperationMode%* = 1,2,5,6 or 7.

It is recommended that this function block be called rapidly and continuously from within the Background task, to detect events as soon as possible after they happen. When an event is ready for the user, the outputs from the function block will indicate so, with Status% being the intended indicator of the event's validity. It is not necessary to delete an event as it is presented once (and only once) to the user by this function block. If more than one event is to be stored, it is essential that the user records each required event e.g. in an array.

Up to 256 events can be stored in the operating system and still be read successfully by using the GetRegistrationEvent function block.

```
(Status%, EventDescriptor%, LeadingPosition%, Distance%,  
TrailingPosition%) = GetRegistrationEvent(ChannelNumber%)
```

## Arguments

ChannelNumber%

0: Capture Channel 0, or SM-Register digital input [0].

1: Capture Channel 1, or SM-Register digital input [1].

## Return values

Status%

See Table 5.5 *Status% Output Table* on page 65.

EventDescriptor%

If Status% returns 1 or REGISTRATION\_TOTAL\_SUCESS:

1: Falling edge has been captured.

2: Rising edge has been captured.

3: Negative pulse has been captured. (Falling edge followed by rising edge).

4: Positive pulse has been captured. (Rising edge followed by falling edge).

If Status% returns 2 or REGISTRATION\_PARTIAL\_SUCESS:

1: Falling edge has been captured.

2: Rising edge has been captured.

3: Negative pulse has been captured. (Falling edge followed by rising edge).

4: Positive pulse has been captured. (Rising edge followed by falling edge).

All other Status% values:

0: Invalid or no event.

LeadingPosition%

The leading edge position at which the event occurred.

Distance%

The width of the pulse if the event is a pulse, or the distance from the previous edge if the event is an edge. The first Distance% after enabling the channel in an edge mode will be zero, as no preceding edge exists.

TrailingPosition%

This field will always be zero in edge detection modes. For pulse detection modes this output reports the position of the trailing edge of the pulse. This enables the position of the trailing edge of the pulse to be known without needing to know the direction of travel and calculate the trailing edge position from the leading edge position and the pulse width.

### 5.4.11 SetRegistrationStartCondition

This function block must be called to define a starting condition before the specified channel can be configured, in an automated registration mode.

```
Status% =  
SetRegistrationStartCondition(StartConditionArray%, ChannelNumber%)
```

Arguments

startConditionArray%

This array allows the user to specify a series of conditions which must be met to constitute a frame start condition:

Number of event specification

Element 0 = Number of events in trigger (up to 8).

Element 1 = Number of events in result (up to 256).

Trigger pulse specification

Element 2 = Minimum distance from the start of the previous pulse.

Element 3 = Maximum distance from the start of the previous pulse.

Element 4 = Minimum pulse width.

Element 5 = Maximum pulse width.

Elements 2 to 5 are repeated for every additional pulse (n) in the trigger

Element  $4n-2$  = Minimum distance from the start of the previous pulse.

Element  $4n-1$  = Maximum distance from the start of the previous pulse.

Element  $4n$  = Minimum pulse width.

Element  $4n+1$  = Maximum pulse width.

**NOTE** The minimum and maximum distance from the previous pulse, is specified in this way to minimise the effect of material stretch when looking for patterns of pulses and gaps e.g. when printing onto plastic film. Other methods of specification like using the distance from the first pulse to specify the pattern, would require progressively wider limits as the end of the pattern is reached.

channelNumber%

0: Capture Channel 0, or SM-Register digital input [0].

1: Capture Channel 1, or SM-Register digital input [1].

Return values

status%

See Table 5.5 *Status% Output Table* on page 65.

5.4.12 GetRegistrationFrame

This function block is used to check for and extract an available registration data frame, in the specified capture channel, when the channel is operating in an automated mode e.g. *OperationMode%* = 3 or 4.

If a frame is available, the data is extracted in one go, and is placed in to the *OutputArray%*.

If a start condition has not been successfully configured this function block will return a -21 *Status%*.

Up to 256 pulses or 64 edges can be stored in each frame.

The SM-Register operating system double-buffers the event information so that whenever this function block is called, the most recently completed frame of data will be passed back to the user program, such that a new frame may be captured while the old one is being read out by the user program.

The system will not store more than one new frame. It is therefore essential to:

- 1) Receive no more than one registration frame in a motion engine period.

2) Call `GetRegistrationFrame` at least once every position update cycle In order that no frames will be missed.

Most frames are likely to consist of many pulses and will by nature take a considerable amount of time to take place; normally there will be considerably more than one motion engine period per frame of data received by the module, so this restriction is unlikely to cause a practical problem.

**NOTE**

If the trigger condition occurs during a frame capture, the first capture will be reset without having completed. In such cases a `Status%` of -25 will be reported. The most likely causes of this are an insufficiently unique trigger condition, or requesting more pulses in the output frame than exist under the repeat distance of the input pulses. If the `OutputArray%` is not large enough to hold all of the data, a `Status%` of 2 is returned, and the function block copies as much data as it can fit into `OutputArray%`.

```
(Status%, NumberOfEvents%) =  
GetRegistrationFrame(OutputArray%, Mode%, ChannelNumber%)
```

### Arguments

`OutputArray%`

The array into which the data should be placed. See `NumberOfEvents%` for contents of `Array%[]`.

`Mode%`

The mode in which to report the frame:

**1 or ABSOLUTE\_MODE.**

The pulse data described using their absolute pulse width and position. The frame will reported as pulses consisting of:

[2n-1] Absolute pulse position.

[2n] Absolute pulse width.

(n is the pulse number, starting from pulse 1)

**2 or FRAME\_RELATIVE\_MODE.**

The pulse positions will be reported relative to the start of the frame, with the exception of the first pulse which will be reported as an absolute position. The frame will reported as pulses consisting of:

[2n-1] Pulse distance from start of frame.

[2n] Absolute pulse width.

(n is the pulse number, starting from pulse 1)

**3 or PULSE\_RELATIVE\_MODE.**

The pulse positions will be reported relative to the previous pulse, with the exception of the first pulse which will be reported as an absolute position. The frame will reported as pulses consisting of:

[2n-1] Pulse distance from start of previous pulse.

[2n] Absolute pulse width.

(n is the pulse number, starting from pulse 1)

**4 or HYBRID\_MODE.**

The pulse positions will be reported in all of the above modes for greatest flexibility. The pulse positions within the first pulse will be reported in absolute mode. The frame will reported as pulses consisting of:

- [4n-3] Absolute pulse position.
  - [4n-2] Pulse distance from start of frame.
  - [4n-1] Pulse distance from start of previous pulse.
  - [4n] Absolute pulse width.
- (n is the pulse number, starting from pulse 1)

ChannelNumber%  
 0: Capture Channel 0, or SM-Register digital input [0]  
 1: Capture Channel 1, or SM-Register digital input [1]

**Return values**

Status%  
 See Table 5.5 *Status% Output Table* on page 65

NumOfEvents%  
 The number of events which have been passed into OutputArray%.  
 Under normal operation NumberOfEvents% will be equal to TriggerCondition%[1].  
 NumberOfEvents% may be lower than TriggerCondition%[1] if OutputArray% is not large enough to hold all pulse descriptions.  
 Each pulse is described by entries in OutputArray% as defined by Mode%.  
 Examples:  
 Mode 1:  
 OutputArray%[0] = Pulse0 position  
 OutputArray%[1] = Pulse0 width  
 OutputArray%[2] = Pulse1 position  
 OutputArray%[3] = Pulse1 width  
 OutputArray%[4] = Pulse2 position  
 OutputArray%[5] = Pulse2 width  
  
 OutputArray%[X] = PulseY position  
  
 In general form:  
 OutputArray%[2n-1] = Pulse n Position  
 OutputArray%[2n] = Pulse n Width  
 (n is the pulse number, starting from pulse 1)

Mode 2:  
 OutputArray%[0] = Pulse0 distance from frame start  
 OutputArray%[1] = Pulse0 width  
 OutputArray%[2] = Pulse1 distance from frame start  
 OutputArray%[3] = Pulse1 width  
 OutputArray%[4] = Pulse2 distance from frame start  
 OutputArray%[5] = Pulse2 width.  
  
 OutputArray%[X] = Pulse Y distance from frame start

In general form:

OutputArray%[2n-1] = Pulse n distance from frame start  
OutputArray%[2n] = Pulse n width  
(n is the pulse number, starting from pulse 1)

Mode 3:

OutputArray%[0] = Pulse0 distance from previous pulse  
OutputArray%[1] = Pulse0 width  
OutputArray%[2] = Pulse1 distance from previous pulse  
OutputArray%[3] = Pulse1 width  
OutputArray%[4] = Pulse2 distance from previous pulse  
OutputArray%[5] = Pulse2 width

OutputArray%[X] = PulseY distance from previous pulse

In general form:

OutputArray%[2n-1] = Pulse n distance from previous pulse  
OutputArray%[2n] = Pulse n width  
(n is the pulse number, starting from pulse 1)

Mode 4:

OutputArray%[0] = Pulse0 position  
OutputArray%[1] = Pulse0 distance from frame start  
OutputArray%[2] = Pulse0 distance from previous pulse  
OutputArray%[3] = Pulse0 width  
OutputArray%[4] = Pulse1 position  
OutputArray%[5] = Pulse1 distance from frame start  
OutputArray%[6] = Pulse1 distance from previous pulse  
OutputArray%[7] = Pulse1 width  
OutputArray%[8] = Pulse2 position  
OutputArray%[9] = Pulse2 distance from frame start  
OutputArray%[10] = Pulse2 distance from previous pulse  
OutputArray%[11] = Pulse2 width

OutputArray%[X] = PulseY position  
OutputArray%[X + 1] = PulseY distance from frame start  
OutputArray%[X + 2] = PulseY distance from previous pulse  
OutputArray%[X + 3] = PulseY width

In general form:

OutputArray%[X] = PulseY position  
OutputArray%[X + 1] = PulseY distance from frame start  
OutputArray%[X + 2] = PulseY distance from previous pulse



OutputArray%[X + 3]   = PulseY width  
(X is the array element number, starting from element 0)

OutputArray%[4n -3]   = Pulse n position  
OutputArray%[4n -2]   = Pulse n distance from frame start  
OutputArray%[4n -1]   = Pulse n distance from previous pulse  
OutputArray%[4n]       = Pulse n width  
(n is the pulse number, starting from pulse 1)

5.4.13 SetRegistrationTrimTimings

This operation will apply delay compensation values to the incoming 24V signal from the registration sensor. The allowable range for rising and falling edges is 0 to 250000ns (0 to 250µs). The real resolution of the trim adjustments is 139ns, so the input values will be rounded to the nearest number of 139ns units.

A module reset (including initial power up) clears the trim timings to zero for rising and falling edges. Enabling, disabling and reconfiguring the channel for which the trim timings have been specified does not affect the timing values.

The registration trim timings can be applied at any time, with the channel configured or unconfigured, enabled or disabled, but the new values will not be used until the next encoder update cycle if the channel is enabled. The changes will take effect immediately if they are made from Initial task, as the registration engine is not running when the Initial task is running.

```
Status% = SetRegistrationTrimTimings(RisingEdgeTrim%, FallingEdgeTrim%,
ChannelNumber%)
```

Arguments

RisingEdgeTrim%  
The time in nanoseconds by which to advance the perceived rising edge. 0 to 250000 nanoseconds.

FallingEdgeTrim%  
The time in nanoseconds by which to advance the perceived falling edge. 0 to 250000 nanoseconds.

ChannelNumber%  
0: Capture Channel 0, or SM-Register digital input [0]  
1: Capture Channel 1, or SM-Register digital input [1]

Return values

status%  
See Table 5.5 *Status% Output Table* on page 65

#### 5.4.14 GetRegistrationFilter

This operation allows the user to determine the status of the registration filter.

##### NOTE

Note that all values returned are the actual values in use at the time of the function block call, and that if the filter is being modified at the time, or fewer than one position update cycle has occurred since the filter values were modified, then one or more of the new filter values might not have been applied yet.

```
(Status%, Configured%, Enabled%, Minimum%, Maximum%) =  
GetRegistrationFilter(ChannelNumber%)
```

#### Arguments

ChannelNumber%

- 0: Capture Channel 0, or SM-Register digital input [0]
- 1: Capture Channel 1, or SM-Register digital input [1]

#### Return values

Status%

See Table 5.5 *Status% Output Table* on page 65

Configured%

- 0: The registration filter is not currently configured.
- 1: The registration filter is currently configured.

Enabled%

- 0: The registration filter is not currently enabled.
  - 1: The registration filter is currently enabled.
- If Configured% returns 0 then Enabled% returns 0.

Minimum%

- The minimum distance configured in the filter.
  - For edge modes this represents the minimum distance between edges.
  - For pulse modes this represents the minimum width of the pulse.
- If Configured% returns 0 then Minimum% returns 0.

Maximum%

- The maximum distance configured in the filter.
  - For edge modes this represents the maximum distance between edges.
  - For pulse modes this represents the maximum width of the pulse.
- If Configured% returns 0 then Maximum% returns 0.

### 5.4.15 GetRegistrationStartCondition

This operation allows the user to determine the current start condition, (if configured), for the automated registration modes.

**NOTE** Any values returned are the actual values configured at the time of the function block call, and that if the start condition is being modified by another task at the time, the contents of the output array may be a mixture of the old and new start condition. If no registration start condition is currently configured, then OutputArray% will not be modified.

Status% = GetRegistrationStartCondition(ChannelNumber%, OutputArray%)

#### Arguments

ChannelNumber%

- 0: Capture Channel 0, or SM-Register digital input [0]
- 1: Capture Channel 1, or SM-Register digital input [1]

OutputArray%

The array into which the start condition will be copied. See Chapter 5.4.11 *SetRegistrationStartCondition* on page 52 for details on the format of the data in OutputArray%

#### Return values

Status%

See Table 5.5 *Status% Output Table* on page 65

### 5.4.16 GetRegistrationStatus

This operation allows the user to determine the general operation status of the specified channel.

(Status%, Configured%, Enabled%, PositionDataSource%, PositionUnits%, PositionFormat%, OperationMode%) = GetRegistrationStatus(ChannelNumber%)

#### Arguments

ChannelNumber%

- 0: Capture Channel 0, or SM-Register digital input [0]
- 1: Capture Channel 1, or SM-Register digital input [1]

OutputArray[%]

The array into which the start condition will be copied.

#### Return values

Status%

See Table 5.5 *Status% Output Table* on page 65

Configured%

- 0: The channel is currently not configured in a valid registration mode.
- 1: The channel is currently configured in a valid registration mode.

Enabled%

Safety Information
Introduction
Installation
Getting started
SM-Register functionality
Parameters
DP-PL Programming
Communications
Freeze and Marker
CT-Sync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

- 0: The channel is currently not enabled.
  - 1: The channel is currently enabled.
- If Configured% returns 0 then Enabled% returns 0

#### PositionDataSource%

If Configured% returns 1:

- 0: The channel is configured to take positions from the reference encoder object.
  - 1: The channel is configured to take positions from the feedback encoder object.
- If Configured% returns 0, the value of PositionDataSource% must be disregarded.

#### PositionUnits%

Positive non-zero values: The number of user units in a revolution, to which all positions should be converted. Reported positions are rounded to the nearest user position unit. Whether reported positions include turns information is defined by PositionFormat%. This is referred to as "Full Scale Mode".

0 to -16: The complement of the number of turns bits to be used for position values. This is referred to as "Turns Bits Scaled Mode"

**NOTE** The reported positions can include turns information or not, and is defined by the *PositionUnits%* input. If turns bits are specified, it is intended that this setting should be the same as the number of turns bits selected in the module which is controlling the registration motion. This enables any position errors / offsets calculated by the SM-Register module to be passed directly to one of the Control Techniques motion platforms e.g. APC (Advanced Position Controller) or PLCopen, without the need for units conversion. If the unit scaling of the motion platform differs from the SM-Register scaling, then a new conversion function will be required.

#### PositionFormat%

0: Turns and position information is reported when reading positions. Positions are always in the range  $-2^{31}$  to  $2^{31}-1$

1: Turns information is omitted when reading positions. Distances may be greater than one turn so distances may exceed the range of positions given below.

Positions are always in the range:

PositionUnits% > 0: 0 to (PositionUnits% - 1)

- PositionUnits% < 0:  $-2^{(32 + \text{PositionUnits}\%)} - 1$  to  $2^{(32 + \text{PositionUnits}\%)}$

**NOTE** If *PositionUnits%* is positive and not a power of 2 ("Full Scale Mode" positioning), e.g. 5000, then turns information may only be used for applications whose total travel is less than  $(2^{32} / \text{PositionUnits}\%)$  revolutions. If this total distance is exceeded a discontinuity will be seen in the reported position. For most rotary applications turns information must NOT be used in "Full Scale Mode" position. Where PositionUnits% is a positive non power of 2, and turns information is selected, the ConfigureRegistrationMode function block will return a Status% of 2, indicating that the axis has been configured, but care must be taken that the axis position does not go past  $2^{31}-1$  in the forwards direction, or past  $-2^{31}$  in the reverse direction.

#### operationMode%

- 1: This channel will capture negative pulses in manual mode.
- 2: This channel will capture positive pulses in manual mode.

- 3: This channel will capture negative pulses in automated mode.
- 4: This channel will capture positive pulses in automated mode.
- 5: This channel will capture negative edges in manual mode.
- 6: This channel will capture positive edges in manual mode.
- 7: This channel will capture both positive and negative edges in manual mode.

### 5.4.17 GetRegistrationTrimTimings

This operation allows the user to determine the present trim timings in use by the specified channel.

The timings reported are the actual timings being used, and may differ slightly to those set using SetRegistrationTrimTimings because of the system timer resolution of 139ns units.

```
(Status%, RisingTrimNanoseconds%, FallingTrimNanoseconds%) =
GetRegistrationTrimTimings(ChannelNumber%)
```

#### Arguments

- ChannelNumber%
- 0: Capture Channel 0, or SM-Register digital input [0]
  - 1: Capture Channel 1, or SM-Register digital input [1]

#### Return values

Status%  
See Table 5.5 *Status% Output Table* on page 65

RisingTrimNanoseconds%  
The delay on rising edges which must be compensated in the module.  
See SetRegistrationTrimTimings for valid range.

FallingTrimNanoseconds%  
The delay on falling edges which must be compensated in the module.  
See SetRegistrationTrimTimings for valid range.

### 5.4.18 GetRegistrationWindow

This operation allows the user to determine the present window in use by the specified channel.

```
(Status%, Configured%, Enabled%, InWindow%, Minimum%, Maximum%, Length%,
RepeatDistance%, RepeatDistanceInUse%) =
GetRegistrationWindow(ChannelNumber%)
```

#### Arguments

- ChannelNumber%
- 0: Capture Channel 0, or SM-Register digital input [0]
  - 1: Capture Channel 1, or SM-Register digital input [1]

#### Return values

Status%  
See Table 5.5 *Status% Output Table* on page 65

#### Configured%

- 0: The registration window is not currently configured.
- 1: The registration window is currently configured.

#### Enabled%

- 0: The registration window is not currently enabled.
  - 1: The registration window is currently enabled.
- If Configured% returns 0 then Enabled% returns 0.

#### InWindow%

- 0: The channel position is not within the window.
  - 1: The channel position is within the window.
- If Enabled% returns 0 then InWindow% returns 0.

#### Minimum%

- The lower window position configured in the channel. (As if specified in the forward direction)
- If Configured% returns 0 then Minimum% returns 0.

#### Maximum%

- The upper window position configured in the channel. (As if specified in the forward direction)
- If Configured% returns 0 then Maximum% returns 0.

#### Length%

- The length of the window configured. (Saves the user having to calculate it)
- If Configured% returns 0 then WindowLength% returns 0.

#### RepeatDistance%

- The repeat distance configured.
- If Configured% returns 0 then RepeatDistance% returns 0.

#### RepeatDistanceInUse%

- Indicates if the repeat distance is currently in use. Under some circumstances a repeat distance may have been specified but may not be suitable for use. This is the easy way for the user to discover if the repeat distance is really in effect.
- If Configured% returns 0 then RepeatDistanceInUse% returns 0.

### 5.4.19 CalculateRegistrationPosition

This function block allows the user to determine the position of a theoretical point or the end of a pulse etc., using a starting position and a distance.

Calculating such positions can be a difficult task if turns information is omitted, as sometimes the width of a pulse cannot be added directly to the start position of a pulse (or subtracted from the start position if rotation is happening in the negative direction) without overflowing the range of representable positions e.g:

If 16 turns bits are in use, and turns information is not being reported, all positions lie in the range 0 to 65535. Therefore a pulse which starts at 65500 and has a width of 100 units will end at position 64, NOT at 65600, which does not exist.

```
(Status%, Position%) =  
CalculateRegistrationPosition(StartPosition%,Distance%,Direction%,  
ChannelNumber%)
```

**Arguments**

**StartPosition%**  
The start position to use as the basis for the calculation.

**Distance%**  
The distance from StartPosition%. Distance% must be positive, and may be greater than 1 revolutions worth of units, even if turns information is not reported in the results.

**Direction%**  
The direction in which Distance% changes from StartPosition%:  
0: Forward  
1: Backward

**ChannelNumber%**  
0: Capture Channel 0, or SM-Register digital input [0]  
1: Capture Channel 1, or SM-Register digital input [1]


**Return values**

**Status%**  
See Table 5.5 *Status% Output Table* on page 65

**Position%**  
The position of the point described by StartPosition%, Distance% and Direction%.

**5.4.20 CalculateRegistrationDistance**

This operation allows the user to determine the distance between two positions; a start position and an end position. Calculating distances is not trivial if turns information is omitted, as sometimes the pair of positions will span the rotary rollover point, meaning that a simple subtraction will not give the correct result.



This function block has the limitation that if the positions do not include turns information, the distance calculated will always be less than one revolution. This might seem obvious but if an event occurred every third revolution on a system in which turns information is not being reported, and those positions were fed into this function the distance would NOT be equal to 3 revolutions - it would be approximately zero as the position within the revolution for each pulse would be approximately identical.

This is the one case where a distance will not exceed one revolution, but that is a consequence of using positions which do not include turns information.

```
(Status%, Distance%) =  
CalculateRegistrationDistance(StartPosition%,EndPosition%,  
ChannelNumber%)
```

## Arguments

StartPosition%

The start position to use as the basis for the calculation.

EndPosition%

The end position to use as the basis for the calculation.

ChannelNumber%

0: Capture Channel 0, or SM-Register digital input [0]

1: Capture Channel 1, or SM-Register digital input [1]

## Return values

Status%

See Table 5.5 *Status% Output Table* on page 65

Distance%

The distance described by StartPosition% and EndPosition%

If the input positions (StartPosition% and EndPosition%) are reversed, the reported distance is calculated for the opposite direction of rotation, e.g:

65536 units per revolution, no turns information.

(Status%, Distance%) = CalculateRegistrationDistance(65500,20,CHANNEL0),

Distance% = 56

(Status%, Distance%) = CalculateRegistrationDistance(20,65500,CHANNEL0),

Distance% = 65480

## 5.4.21 GetRegistrationPosition

This operation allows the user to determine the position of the encoder to which the channel is attached, converted to the specified user units and turns mode, without the need to “see” a registration event.

```
(Status%, Position%) = GetRegistrationPosition(ChannelNumber%)
```

## Arguments

ChannelNumber%

0: Capture Channel 0, or SM-Register digital input [0]

1: Capture Channel 1, or SM-Register digital input [1]

## Return values

Status%

See Table 5.5 *Status% Output Table* on page 65

Position%

The position of the encoder at the moment (subject to encoder update rate) in the units of the specified channel.



## 5.4.22 Status% Output

All registration function blocks return a Status% indication to inform the user of the result of the attempted operation. For ease of use, this output will always be from the following list of values, though not all function blocks may be capable of generating all output values.

**Table 5.5 Status% Output Table**

Status%	Definition	Description
-99	Failure	REGISTRATION_UNDEFINED_STATUS A catch-all error condition. If this error is reported contact your Control Techniques support centre.
-29	Failure	REGISTRATION_WINDOW_LENGTH_GREATER_THAN_REPEAT_DISTANCE It is not permissible to have a window length which is greater than the repeat distance.
-28	Failure	REGISTRATION_ILLEGAL_NUMBER_REPORTED_PULSES The specified number of pulses to report is invalid.
-27	Failure	REGISTRATION_ILLEGAL_NUMBER_TRIGGER_PULSES The specified number of trigger pulses in the start condition is invalid.
-26	Failure	REGISTRATION_INVALID_WINDOW_LENGTH The window length is invalid.
-25	Failure	REGISTRATION_FRAME_CAPTURE_ABORTED A duplicate trigger condition was seen during frame capture which reset the frame. Thus a valid trigger condition was not reported due to no complete frame being available. (Most likely the frame reporting length is too long or the trigger condition is not sufficiently unique.)
-24	Failure	REGISTRATION_ILLEGAL_REPORT_MODE The reporting mode specified is not valid.
-23	Failure	REGISTRATION_INVALID_DISTANCE The distance specified is not valid. (Distances cannot be negative.)
-22	Failure	REGISTRATION_INVALID_POINTER An invalid pointer was specified.
-21	Failure	REGISTRATION_NO_TRIGGER_CONDITION No trigger condition has been defined for automated register mode to seek.
-20	Failure	REGISTRATION_INVALID_DIRECTION The specified direction was not valid.
-19	Failure	REGISTRATION_NOT_ENABLED The specified feature must be enabled for this operation to complete successfully. (Feature is already disabled.)
-18	Failure	REGISTRATION_INVALID_OFFSET The specified offset is not valid.
-17	Failure	REGISTRATION_INVALID_PULSE_INTERVAL The specified pulse interval is not valid.
-16	Failure	REGISTRATION_ILLEGAL_MODE The specified mode is not valid, or the operation cannot be completed in the present mode.

Safety Information
Introduction
Installation
Getting started
<b>SM-Register Functionality</b>
Parameters
DP_L
Communications
Freeze and Marker
CTSync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

Status%	Definition	Description
-15	Failure	REGISTRATION_CURRENTLY_ENABLED The operation could not complete because the channel or specified feature of the channel is already / currently enabled.
-14	Failure	REGISTRATION_INVALID_TRIM_TIMING The timing adjustment value was out of range.
-13	Failure	REGISTRATION_INVALID_REPORT_LENGTH The number of elements requested in automated mode cannot be provided.
-12	Failure	REGISTRATION_INVALID_COMPARE_LENGTH This number of pulses cannot be used as a trigger pattern.
-11	Failure	REGISTRATION_INVALID_ARRAY_LENGTH The specified array was too small to contain sufficient data.
-10	Failure	REGISTRATION_INVALID_ENCODER_UPDATE_PERIOD The registration system may only be run with encoder update rate (#81.16) set to 0 or 1.
-9	Failure	REGISTRATION_INVALID_INPUT One or more inputs was invalid. (e.g. position mode or position format.)
-8	Failure	REGISTRATION_CHANNEL_DOES_NOT_EXIST The specified channel does not exist on this module.
-7	Failure	REGISTRATION_INCOMPATIBLE_MODE The channel is not in a suitable mode for the requested operation.
-6	Failure	REGISTRATION_NO_WINDOW The window could not be set.
-5	Failure	REGISTRATION_INVALID_ENCODER_OBJECT The specified encoder object could not be used.
-4	Failure	REGISTRATION_INVALID_PULSE_WIDTH The specified pulse width was not valid.
-3	Failure	REGISTRATION_NOT_CONFIGURED The operation could not complete because the registration system or feature is not configured correctly.
-2	Failure	REGISTRATION_INVALID_POSITION_MODE The specified position mode is invalid.
-1	Failure	REGISTRATION_INVALID_POSITION The specified position is not valid.
0	Failure	REGISTRATION_NO_DATA_AVAILABLE No data available.
1	Success	REGISTRATION_TOTAL_SUCCESS Operation completed successfully. If the operation was to extract data, the data has been placed at the destination and is valid.
2	Success (but some operations may not have been completed)	REGISTRATION_PARTIAL_SUCCESS Operation completed successfully, but due to a constraint not all of the information requested has been provided.

## 6 Parameters

### 6.1 Overview

The SM-Register contains two parameter databases:

- The drive database:  
This contains the entire drive parameter set. The SM-Register caches this database in its own non-volatile *flash* memory. At power-up the module will check to see if this cache matches that of the drive. If it doesn't the database will be loaded from the drive, during which time the word "Loading" will appear for a few seconds on the drive display. This will not occur again unless the SM-Register is moved to a different drive with different firmware or the drive firmware is updated.
- The SM-Register database:  
This database contains all parameters held locally to the module such as PLC registers as well as any other short-cut parameters (menus 90, 91, etc.)

### 6.2 Saving parameters

There are different ways of saving parameters depending on the type of parameter that needs to be saved. These are explained in detail in the following sections:

#### 6.2.1 Saving SM-Register parameters

The parameters that are saved to the SM-Register when performing the actions shown below are:

- Menus 70, 71, 74 and 75 (equivalent to P, Q, T and U register sets)
- Menu 20

To save the parameters on demand:

1. Set **Pr 81.19** to 1 (**Save Request**)
2. Press the reset button  
Parameter **Pr 81.19** will reset to zero automatically and the module and drive will be reset.

To save the parameters on Under Voltage (UU):

- Set **Pr 81.20** to 1 (**Enable "UU trip" Save**)

Note that simply performing the above operations will not save menu 20. To save menu 20 you will need to perform the above operations but ensure that parameter **Pr 81.21 (Enable menu 20 save and restore)** is set to 1 before doing so. This parameter does not need a module reset for the changes to become active.

#### 6.2.2 Restoring menu 20 parameters

To restore menu 20 parameters on power-up, parameter **Pr 81.21 (Enable menu 20 save and restore)** needs to be at 1 at power-up therefore a drive parameter save is required. See section 6.2.3 *Saving drive parameters*.

#### 6.2.3 Saving drive parameters

The parameters that are saved to the drive when performing the actions shown below are:

- Menus 1 thru 14, 18, 19, 21 and 22.
- Menus 15, 16 and 17, if a module is present in the relevant slot.

To save the drive parameters:

1. Set **Pr 0.00**=1000 (parameter zero in any menu when using drive keypad)
2. Set **Pr 10.38**=100 (simulates pressing the reset button on the drive keypad)

## 6.3 Configuration parameters

The basic configuration (or setup) parameters are held in the appropriate menu for the slot where the SM-Register is installed.

Slot	Menu
1	15
2	16
3	17

In addition to these menus, an alias of the appropriate menu is available as local menu 81 within the SM-Register. This menu can be accessed from the user DPL program or via communications (CTNet/CT-RTU/EIA-485) and provides a convenient way to read or change the setup parameters without having to know which slot the SM-Register is installed in.



Unless otherwise indicated, these parameters are only read when the SM-Register is first powered up, on a reset or on a *REINIT* DPL command. Changing one of these parameters on the fly will have no immediate effect.

To reset the SM-Register from the drive display, enter the value of 1070 in parameter zero of any menu and press the reset button.

### NOTE

Throughout this User Guide, the configuration parameters will be referred to as Pr **81.XX**. When setting parameters directly on the drive keypad use the appropriate menu 15, 16 or 17 instead.

### NOTE

The update rate specified for any parameter refers to the rate at which the parameter is updated for reading or when writing, when the new value takes effect. "Initialisation" means that the parameter is read only on module reset or REINIT DPL command.



Changing the drive mode will clear all configuration and application parameters back to their default value as well as drive parameters. This can be avoided by using the code **1255** in parameter zero rather than the usual **1253**. Only drive parameters will be defaulted, and menus 15 to 20 will be left unchanged.

### 6.3.1 Parameter descriptions

Pr 81.01	Module Code		
Access	RO	Range	306
Default	N/A	Update Rate	N/A

The Module Code indicates the type of module that is installed in the corresponding slot. For SM-Register the module code is 306.

Pr 81.02	Firmware Version - Major		
Access	RO	Range	00.00 to 99.99
Default	N/A	Update Rate	N/A

Specifies the major revision number of the operating system of the SM-Register. Use in conjunction with Pr 81.51 to form the complete version number

Pr 81.03	DPL Program Status		
Access	RO	Range	0 to 3
Default	0	Update Rate	1ms of change

Provides the run status of the user DPL program in the SM-Register. The following values are defined:

Display	Value	Description
nonE	0	No DPL program present
StoP	1	DPL program is stopped
run	2	DPL program is running
triP	3	Run-time error. ERROR task running or DPL program stopped

Pr 81.04	Available System Resource		
Access	RO	Range	0 to 100
Default	N/A	Update Rate	200ms

Displays the free CPU resource as a percentage of the current background execution time calculated over 200ms.

Pr 81.05	EIA-485 Address		
Access	RW	Range	0 to 255
Default	11	Update Rate	Initialisation

Defines the address of this node for ANSI and Modbus communications protocols.

For the ANSI protocol the address range is 11 to 99 where the first digit is the group address and the second digit is the unit number. Both digits must be in the range of 1-9. Zero is not permitted since it is used by the master for addressing groups of nodes.

This parameter has no effect if the EIA-485 mode is 25 (CTSync) or 26 (CTSync).

<b>Pr 81.06</b>	<b>EIA-485 Mode</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 255
<b>Default</b>	1	<b>Update Rate</b>	Initialisation

Defines the mode of operation (or protocol) for the on-board EIA-485 port. For details of these modes, refer to Table 6.1 *Serial modes - Pr 81.06* ..

**Table 6.1 Serial modes - Pr 81.06**

Mode	Description
1	4-wire CT-ANSI Slave The port is set to 1 start bit, 7 data bits, even parity and 1 stop bit.
2	Reserved.
3	Reserved.
4	Reserved
5	2-wire CT-ANSI Slave The port is set to 1 start bit, 7 data bits, even parity and 1 stop bit.
6	User mode. 1 start bit, 7 data bits, EVEN parity, 1 stop bit (10 bits total).
7	User mode. 1 start bit, 8 data bits, EVEN parity, 1 stop bit (11 bits total).
8	User mode. 1 start bit, 8 data bits, NO parity, 1 stop bit (10 bits total).
9	Reserved.
10	Reserved.
11	Reserved.
12	Reserved.
13,43,73	4-wire Modbus RTU slave The EIA-485 port is set for: Mode 13: 1 start bit, 8 data bits, NO parity, 2 stop bits. Mode 43: 1 start bit, 8 data bits, EVEN parity, 1 stop bit. Mode 73: 1 start bit, 8 data bits, ODD parity, 1 stop bit.
14, 44, 74	4-wire Modbus ASCII slave The EIA-485 is set for: Mode 14: 1 start bit, 7 data bits, NO parity, 2 stop bits. Mode 44: 1 start bit, 7 data bits, EVEN parity, 1 stop bit. Mode 74: 1 start bit, 7 data bits, ODD parity, 1 stop bit.
15, 45, 75	2-wire Modbus RTU slave The EIA-485 port is set for: Mode 15: 1 start bit, 8 data bits, NO parity, 2 stop bits. Mode 45: 1 start bit, 8 data bits, EVEN parity, 1 stop bit. Mode 75: 1 start bit, 8 data bits, ODD parity, 1 stop bit.
16, 46, 76	2-wire Modbus ASCII slave The EIA-485 is set for: Mode 16: 1 start bit, 7 data bits, NO parity, 2 stop bits. Mode 46: 1 start bit, 7 data bits, EVEN parity, 1 stop bit. Mode 76: 1 start bit, 7 data bits, ODD parity, 1 stop bit.
17, 47, 77	4-wire Modbus RTU master The EIA-485 port is set for: Mode 17: 1 start bit, 8 data bits, NO parity, 2 stop bits. Mode 47: 1 start bit, 8 data bits, EVEN parity, 1 stop bit. Mode 77: 1 start bit, 8 data bits, ODD parity, 1 stop bit.
18, 48, 78	4-wire Modbus ASCII master The EIA-485 is set for: Mode 18: 1 start bit, 7 data bits, NO parity, 2 stop bits. Mode 48: 1 start bit, 7 data bits, EVEN parity, 1 stop bit. Mode 78: 1 start bit, 7 data bits, ODD parity, 1 stop bit.

**Table 6.1 Serial modes - Pr 81.06**

Mode	Description
19, 49, 79	2-wire Modbus RTU master The EIA-485 port is set for: Mode 19: 1 start bit, 8 data bits, NO parity, 2 stop bits. Mode 49: 1 start bit, 8 data bits, EVEN parity, 1 stop bit. Mode 79: 1 start bit, 8 data bits, ODD parity, 1 stop bit.
20, 50, 80	2-wire Modbus ASCII master The EIA-485 is set for: Mode 20: 1 start bit, 7 data bits, NO parity, 2 stop bits. Mode 50: 1 start bit, 7 data bits, EVEN parity, 1 stop bit. Mode 80: 1 start bit, 7 data bits, ODD parity, 1 stop bit.
25	SM-Register Module CT-Sync Master. The baud rate is fixed at 896875bps.
26	SM-Register Module CT-Sync Slave. The baud rate is fixed at 896875bps.

Pr 81.07	EIA-485 Baud Rate		
Access	RW	Range	0-9 (300-115200bps)
Default	4 (4800)	Update Rate	Initialisation

Defines the baud-rate (or bits-per-second) for the on-board EIA-485 port. The following are supported:

Note that this parameter is not relevant when the EIA-485 port mode is set to 25 (CTSync Master) or 26 (CTSync Slave).

Display	Value	Desc	Display	Value	Desc
300	0	300bps	9600	5	9600bps
600	1	600bps	19200	6	19200bps
1200	2	1200bps	38400	7	38400bps
2400	3	2400bps	57600	8	57600bps
4800	4	4800bps	115200	9	115200bps

Pr 81.08	EIA-485 Turn-around delay		
Access	RW	Range	0 to 255ms
Default	2ms	Update Rate	Initialisation

Defines a fixed delay between receiving a message on the EIA-485 port and the response being transmitted. This can be useful in 2-wire configurations where it takes a finite time for the master (host) to switch from transmit mode to receive mode. There is always at least a 1ms delay and this parameter can be used to extend it.

Pr 81.09	EIA-485 Tx enable delay		
Access	RW	Range	0 to 1ms
Default	0ms	Update Rate	Initialisation

This parameter allows a 1ms delay to be introduced between the SM-Register enabling the EIA-485 transmitter and actually commencing the transmission. This should only be required if it is found that the recipient of the transmission is receiving a corrupted start of message.

<b>Pr 81.10</b>	<b>DPL print routing</b>		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	0	<b>Update Rate</b>	Initialisation

Controls where the output of the DPL PRINT command is sent. If set to zero (Off), the output is sent to the programming client (SyPTPro) and if set to 1 (On) it will be sent to the EIA-485 port.

<b>Pr 81.11</b>	<b>Clock task scheduling (ms)</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 200ms
<b>Default</b>	10ms	<b>Update Rate</b>	Initialisation

Defines the scheduling period (tick-time), in milliseconds, for the DPL CLOCK task. A value of zero will disable the CLOCK task.

**NOTE** Prior to version 01.05.00 Unidrive SP the default for this parameter was 0ms (disabled).

<b>Pr 81.12</b>	<b>POS task scheduling rate</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 6
<b>Default</b>	0	<b>Update Rate</b>	Initialisation

Defines the scheduling rate for the POS tasks to suit the application performance and the resource needed to run the user DPL program. The following values are defined::

Display	Value	Description
diSAbled	0	Disabled
0.25	1	250µs
0.5	2	500µs
1	3	1ms
2	4	2ms
4	5	4ms
8	6	8ms

Set this parameter in order for the user DPL program to automatically run at power-on/ reset. If this is changed and the new setting needs to be used on power-up ensure that a **drive** parameter save is performed.

**NOTE** With the Unidrive SP version 01.03.00 and earlier, the display will not show the actual rate, but its aliased number. For instance 250µs will be shown on the display as 1.

<b>Pr 81.13</b>	<b>Auto-run enable</b>		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	1	<b>Update Rate</b>	Initialisation

<b>Pr 81.14</b>	<b>Global run-time trip enable</b>		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	0	<b>Update Rate</b>	Initialisation

Setting this parameter to 1 will cause the drive to trip when certain run-time errors occur within the SM-Register user DPL program.

For more information, see section 12.1 *Run-time errors* on page 139.



Pr 81.15	Disable reset on trip cleared		
Access	RW	Range	0/1
Default	0	Update Rate	Initialisation

When this parameter is 0, the module will be reset when a drive trip is cleared. When set to 1 the module will be unaffected by a drive trip reset (i.e. continue running).

Pr 81.16	Encoder data update rate		
Access	RW	Range	0-3
Default	0	Update Rate	Initialisation

Display	Description
0	APC data and menu 90 encoder parameters are updated every 250 $\mu$ s.
1	APC data and menu 90 encoder parameters are updated immediately prior to every POS task.
2	APC data and menu 90 encoder parameters are updated immediately prior to every CLOCK task.
3	APC data and menu 90 encoder parameters are never updated. If these are never updated, more processor resource will become free.

Pr 81.17	Enable parameter over-range trips		
Access	RW	Range	0/1
Default	0	Update Rate	Initialisation

Defines the action taken if a user DPL program attempts to write an out of range value to a parameter. When set at 1, a run-time trip will occur (number 44); when set at zero the value will automatically be limited to the maximum/minimum of that parameter.

Pr 81.18	Watchdog enable		
Access	RW	Range	0/1
Default	0	Update Rate	Initialisation

When set it enables the DPL program watchdog function. The DPL WDOG command must then be executed every 200ms. This can be used to protect the program against malfunction. If the command is not executed within a 200ms time period the drive will trip on **SLx.tO**. Please note that the WDOG command must also be executed once for the watchdog to be enabled. This is normally executed at the end of the Initial task.

Pr 81.19	Save request		
Access	RW	Range	0/1
Default	0	Update Rate	100ms

Setting this parameter to 1 will initiate an immediate save of all non-volatile SM-Register data. This consists of the P/Q/T/U PLC register sets and optionally menu 20 (depending upon the setting of Pr 81.21).

**NOTE** Note that this will also cause a reset of the module and this parameter will return back to zero automatically. Also if the drive is tripped, it will be reset. Menu 81 will not be saved.

<b>Pr 81.20</b>	<b>Enable "UU trip" save</b>		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	0	<b>Update Rate</b>	Immediate

Setting this parameter to 1 signals that all non-volatile data of the SM-Register will be automatically saved upon an under voltage (UU) trip of the drive.

**NOTE** Note that when a 'UU' save occurs the SM-Register will be reset.

<b>Pr 81.21</b>	<b>Enable menu 20 save and restore</b>		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	0	<b>Update Rate</b>	Immediate

If set to 1, menu 20 will be saved/restored along with other non-volatile parameters upon a save request (Pr **xx.19**=1) or power-down save (Pr **xx.20**=1). If menu 20 is to be restored on power-up the user must ensure that this parameter is saved in the drive before powering down.

Since menu 20 is a global drive menu, only one option installed to the drive should be used to store and restore menu 20, therefore if more than one SM-Register, SM-Applications or Motion Processor is installed to the drive **only one** should have this parameter set otherwise menu 20 will not be restored correctly on power-up.

**NOTE** Unlike other setup parameters, parameters Pr **81.20** and Pr **82.21** are **not cached**, which means a change to the parameter takes immediate effect.

<b>Pr 81.22</b>	<b>CTNet token ring ID</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 255
<b>Default</b>	0	<b>Update Rate</b>	Initialisation

This parameter allows the user to specify the identity of the CTNet token ring to which a SM-Register is connected. In a system incorporating a single token ring this parameter can be left at its default value. In a system incorporating multiple token rings, separate ID's should be set for each ring. The combination of CTNet Token Ring ID and CTNet node address should be unique

<b>Pr 81.23</b>	<b>CTNet node address</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 255
<b>Default</b>	0	<b>Update Rate</b>	Initialisation

Defines the node address for CTNet. Every node on a CTNet network must have a unique address. Setting this to zero will disable CTNet on this node

<b>Pr 81.24</b>	<b>CTNet baud rate</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 3
<b>Default</b>	1(2.5)	<b>Update Rate</b>	Initialisation

Specifies the data rate for CTNet. All nodes on the network must be set to the same data-rate. The rates are defined as follows

Display	Value	Description	Display	Value	Description
5.000	0	5Mbit/s	1.250	2	1.25Mbit/s
2.500	1	2.5Mbit/s	0.625	3	625kbit/s

<b>Pr 81.25</b>	<b>CTNet sync setup</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 9999
<b>Default</b>	0	<b>Update Rate</b>	Initialisation

Specifies the synchronisation message generation rate for CTNet. This message is used to tell all nodes when to transmit cyclic data. Only one node on the CTNet network should have this parameter set. The format of the update parameter is SSFF, where FF defines the Fast Cyclic data channel update rate, and SS defines the slow cyclic data rate in multiples of FF. So if the parameter value is 1510, fast cyclic data is set every 10ms and slow every 150ms. When using easy mode (see below) it is only necessary to set up the FF (fast cyclic rate).

<b>Pr 81.26, Pr 81.28, Pr 81.30</b>	<b>CTNet easy mode setup parameters</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 25503
<b>Default</b>	0	<b>Update Rate</b>	Initialisation

<b>Pr 81.27, Pr 81.29, Pr 81.31- Pr 81.34</b>	<b>CTNet easy mode setup parameters</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 9999
<b>Default</b>	0	<b>Update Rate</b>	Initialisation

These parameters define the source and destinations for CTNet easy-mode cyclic data.

Parameter	Format	Channel	Description
Pr 81.26	NNSS	1	Defines the destination node number and slot NNN = Node number (0-255). SS = Slot number (1-3). e.g. A value of 201 means node ID 2, slot 1.
Pr 81.27	MMPP	1	Defines the source drive parameter which is to be transmitted MM = Menu number. PP = Parameter number. e.g. A value of 302 means <b>Pr 3.02</b> (speed)
Pr 81.28	NNSS	2	Destination node number and slot for channel 2.
Pr 81.29	MMPP	2	Source drive parameter for channel 2.
Pr 81.30	NNSS	3	Destination node number and slot for channel 3.
Pr 81.31	MMPP	3	Source drive parameter for channel 3.
Pr 81.32	MMPP	1	Slot 1 destination parameter for incoming data.
Pr 81.33	MMPP	2	Slot 2 destination parameter for incoming data.
Pr 81.34	MMPP	3	Slot 3 destination parameter for incoming data.

<b>Pr 81.35</b>	<b>CTNet sync. event task ID</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 4
<b>Default</b>	0	<b>Update Rate</b>	Initialisation

Identifies which of the EVENT tasks will be scheduled when a CTNet synchronization message is received or generated. This sync. is generated by a *master* node (which can be this node) on the CTNet network at a fixed time-base. The following values are defined:

Display	Value	Description	Display	Value	Description
Disabled	0	No event task scheduled	Event2	3	EVENT2 task scheduled
Event	1	EVENT task scheduled	Event3	4	EVENT3 task scheduled
Event1	2	EVENT1 task scheduled			

Pr 81.36	CTNet diagnostics			
Access	RO	Range	-3 to 32767	
Default	N/A	Update Rate	1 second	

The status of the CTNet network is displayed in the CTNet Diagnostic parameter. When the SM-Register is communicating successfully on the CTNet network the number of messages per second is displayed.

**Table 6.2 CTNet diagnostics**

#MM.36	Status	Description
>0	Network OK	Indicates the number of messages per second be processed every second.
0	Network OK, No Data Transfer	The low-level token ring has been established and is active, but the node is not receiving any CTNet data messages.
-1	RECON	A network reconfiguration has been detected.
-2	Initialisation Error	The SM-Register Module was unable to configure the CTNet interface. Check that the node address and data rate are set correctly.
-3	MYRECON	The SM-Register Module forced a CTNet network reconfiguration.

Pr 81.37	Reject download if drive enabled		
Access	RW	Range	0/1
Default	0	Update Rate	Initialisation

If this parameter is set, then if the user attempts to download a new user DPL program or operating system to this module and the drive is enabled the download will be rejected and a run-time trip 70 will occur, if the global run-time trip parameter (Pr 81.14) is set.

Since downloading stops normal operations of the module it may be considered unsafe to do this if the drive system is running, therefore setting this parameter will prevent downloading under this condition.

Pr 81.38	APC run-time trip		
Access	RW	Range	0/1
Default	0	Update Rate	Initialisation

When this parameter is 0 the drive will trip with runtime error 81 if an APC non-recoverable error occurs, such as use of an uninitialised CAM function. When this parameter is 1 the drive will not trip when an APC non-recoverable error occurs.

Pr 81.39	Inter-module drive sync status		
Access	RW	Range	0/1
Default	0	Update Rate	NA

This parameter displays the current module's synchronisation status.

Synchronisation Status	Status
0	The synchronisation master request is zero or another option module is synchronisation master.
1	The option module is synchronisation master.
3	The option module is synchronisation master, but the synchronisation frequency is out of specification or not present.

Pr 81.41	Indexer control		
Access	RW	Range	0/3
Default	0	Update Rate	NA

This is used to control the motion sequence user program.

Value	Status	Value	Status
0	Run	2	Pause
1	Stop	3	Step

Pr 81.42	Pass freeze through to drive		
Access	RW	Range	0/1
Default	0	Update Rate	Initialisation

When this parameter is ON (1), the voltage on the SM-Register digital input 0 (zero) is passed through to the drive's internal Freeze line. This can be seen by other classes of SM-Solutions modules. For further information on the Freeze Input refer to Chapter 9 *Freeze and Marker* on page 122.

Pr 81.43	Freeze invert		
Access	RW	Range	0/1
Default	0	Update Rate	Initialisation

When this parameter is set to zero a freeze occurs on the rising edge of the module's DIGIN0. When it is set to 1 a freeze occurs on the falling edge of the module's DIGIN0. For further information on the Freeze Input refer to Chapter 9 *Freeze and Marker* on page 122.

When this parameter is set to zero the module will freeze data when it sees a rising edge of a freeze pulse from the drive. When it is set to a 1 the module will freeze data when it sees a falling edge of a freeze pulse from the drive. For further information on the Freeze Input refer to Chapter 9 *Freeze and Marker* on page 122.

Pr 81.44	Task priority level		
Access	RW	Range	0 to 255
Default	0	Update Rate	Initialisation

The priority levels of different tasks may be changed with this parameter. The parameter is accessed in a bit-wise manner:

Bit	Value	Meaning
0	0	CTNet task priority is higher than Pos tasks priority.
	1	CTNet task priority is lower than Pos tasks priority. This will reduce the jitter of the POS tasks but could lead to the CTNet task being starved.
1	0	Inter-option communication task priority is higher than the POS tasks.
	1	Inter option communication task priority lower than the POS tasks.
2	0	Turbo CTNet Disabled.
	1	Turbo CTNet Enabled.

Pr 81.45	User set-up parameter 1		
Access	RO	Range	Unsigned 16 bit
Default	0	Update Rate	N/A

This is a user set-up parameter and is read only through the Keypad. It can be written to in DPL code using the WriteUserSetupParameter1 command and read in DPL code using the ReadUserSetupParameter1 command.

Pr 81.46	User set-up parameter 2		
Access	RO	Range	Signed 16 bit
Default	0	Update Rate	N/A

This is a user set-up parameter and is read only through the Keypad. It can be written to in DPL code using the WriteUserSetupParameter2 command and read in DPL code using the ReadUserSetupParameter2 command.

Pr 81.47	User set-up parameter 3		
Access	RO	Range	Unsigned 8 bit
Default	0	Update Rate	N/A

This is a user set-up parameter and is read only through the Keypad. It can be written to in DPL code using the WriteUserSetupParameter3 command and read in DPL code using the ReadUserSetupParameter3 command.

Pr 81.48	Line number of error		
Access	RO	Range	32 bit
Default	0	Update Rate	On error

Specifies the DPL program line number that caused a run-time error. This is valid only when:

- The user program has been compiled with the *debug* option set.
- The error is one that can be generated by user code, for example divide by zero (50) or parameter does not exist (41).

If both of these conditions are not met, the line number parameter will display zero (0).

<b>Pr 81.49</b>	<b>User program ID</b>		
<b>Access</b>	RO/RW	<b>Range</b>	Signed 16-bit
<b>Default</b>	0	<b>Update Rate</b>	See Note

This parameter is available for the user to put in an ID code of their program. This may, for example, be the software version number. Use the function block SETUSERID() to write to this parameter.

<b>Pr 81.50</b>	<b>Run-time Error Code</b>		
<b>Access</b>	RO	<b>Range</b>	0 to 255
<b>Default</b>	0	<b>Update Rate</b>	On error

When a run-time error occurs the error number is placed into this parameter.

See Chapter 12.1 *Run-time errors* on page 139 for further information.

<b>Pr 81.51</b>	<b>Firmware - minor version</b>		
<b>Access</b>	RO	<b>Range</b>	0 to 99
<b>Default</b>	N/A	<b>Update Rate</b>	N/A

Specifies the minor revision number of the operating system of the SM-Register. Use in conjunction with Pr **81.02** to form the complete version number.

## 6.4 Menus 70-75 - PLC registers

These menus provide access to the PLC registers. The PLC registers are Signed 32-bit integers available for user programs and CTNet communications.

The PLC registers are split into 6 sets of 100 parameters numbered 00 to 99. The registers can also be accessed from within a user DPL program by a special variable name or array name.

Menu Number	Access	DPL variable (x=register number)	Number of Registers	Description
70	RW	_Px%, _P%[x]	100	General purpose. Saveable.
71	RW	_Qx%, _Q%[x]	100	General purpose. Saveable.
72	RW	_Rx%, _R%[x]	100	Used for outgoing CTNet cyclic data links. Non-saveable.
73	RW	_Sx%, _S%[x]	100	Used to incoming CTNet cyclic data links. Non-saveable.
74	RW	_Tx%, _T%[x]	100	General purpose. Saveable.
75	RW	_Ux%, _U%[x]	100	General purpose. Saveable.

You will see from the above table that each parameter within menus 70 to 75 has an equivalent DPL variable. This means that you can use either format for accessing a parameter within these menus.

e.g. **Pr 72.01=1** will do the same as **\_R01%=1**, **Pr 75.65=66** will do the same as **\_U65%=66** etc.

Menus 70, 71, 74 and 75 can all be saved into the non-volatile flash memory upon request or automatically when the drive goes into under-voltage (refer to section 6.2 *Saving parameters* on page 67 for more information).

Menus 72 and 73 are used for CTNet cyclic data transfer but if this feature is not being used the registers may be used for any other purpose. However this should be avoided if possible in case cyclic data is used in the future.

Parameters #72.79 and #73.79 will be reserved if AUTOSYNC is in use.

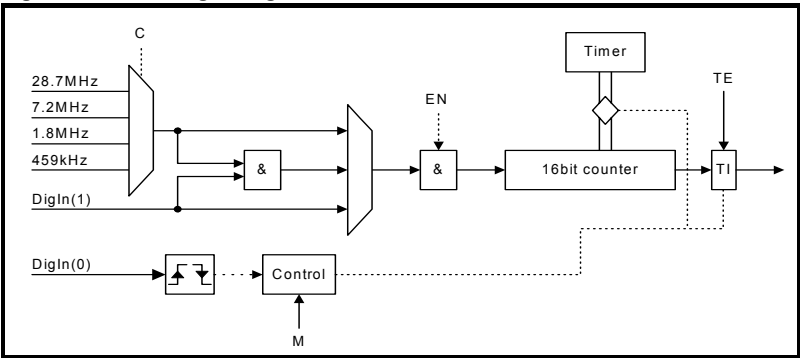
Parameters #71.80 thru #71.99 may be used for the RAM File Recording.

## 6.5 Menu 85 - Timer function parameters

A hardware/counter is built into the SM-Register which has the following features:

- A 16 bit incremental counter.
- Count rate selectable from the internal clock. The clock rate divisor is selectable from 1, 4, 16, 64.
- Count rate selectable from an external clock via the DIGIN1 digital input. The maximum clock rate is 600kHz.
- The timer can be used to schedule one of the 4 DPL Event tasks on wrap-around or an input capture on DIGIN1.
- Counter overflow can be selected by the user up to the full 16 bit range available for the counter.
- The timer can be set to cache the count on a DIGIN0 rising or falling edge transition.

**Figure 6-1 Timer logic diagram**



Pr 85.01	Timer unit control word		
Access	RW	Range	13 bit
Default	N/A	Update Rate	Immediate

**Table 6.3 Control word - Pr 85.01**

Bit	Symbol	Function
b0-b2	TE	Timer EVENT task schedule when the TI flag is set: 0=No Event task scheduled. 1=Schedule Event task. 2=Schedule Event1 task. 3=Schedule Event2 task. 4=Schedule Event3 task.
b3	EN	Enable Timer: 0=Timer is disabled. 1=Timer is enabled.
b4	CS	Clock source: 0=internal clock 1=external clock provided on DIGIN1.
b5-b6	C	Internal clock pre-scale select (ignored if external clock selected): 0=rate /1 (28.7MHz). 1=rate /4. 2=rate /16. 3=rate /64.



**Table 6.3 Control word - Pr 85.01**

Bit	Symbol	Function
b7-b8	M	<p>Timer mode:</p> <p><u>0=Free Running mode</u> The selected clock drives the counter. The TI flag is set on wrap-around.</p> <p><u>1=Capture mode 1</u> The selected clock drives the counter. A rising edge transition on DIGIN0 causes the current counter value to be latched into the TIMER CAPTURE CACHE parameter and the TI flag is set. The counter then continues incrementing (TI is not set on wrap-around).</p> <p><u>2=Capture mode 2</u> The selected clock drives the counter. A falling edge transition on DIGIN0 causes the current counter value to be latched into the TIMER CAPTURE CACHE parameter and the TI flag is set. The counter then continues incrementing (TI is not set on wrap-around).</p>

Pr 85.02	Timer unit status word		
Access	RO	Range	0 to 3
Default	N/A	Update Rate	Immediate

**Table 6.4 Status word - Pr 85.02**


Bit	Symbol	Function
b0	TI	<p>Timer event flag:</p> <p>0=No event has occurred 1=An event has occurred (see description for Pr 85.01) <b>Note:</b> The TI bit is automatically cleared by the operating system if TE flag in Pr 85.01 has a non-zero value. Otherwise it is cleared when the status word is read.</p>
b1	OV	<p>Wrap-around flag</p> <p>0=Wrap-around has not occurred 1=Counter wrap-around has occurred <b>Note:</b> This flag is valid for ALL timer modes and is automatically cleared when the status register is read.</p>

Pr 85.03	Timer unit 16-bit timer count		
Access	RW	Range	16 bit
Default	N/A	Update Rate	Immediate

The current timer value can be read and written at any time using this parameter.

Pr 85.04	Timer unit wrap-around limit		
Access	RW	Range	16 bit
Default	N/A	Update Rate	Immediate

This parameter specifies the value at which Pr 85.03 will wrap-around.



Setting a small wrap-around figure with the timer operating at fast rates, could result in the SM-Register appearing to lock up. This is caused by the wrap-around interrupt being continually serviced and starving the other product features of processor resource. If this occurs the user can reset the SM-Register by performing a 1070 reset on the drive. If a DPL program sets the figures for the timer then, before resetting, the program auto-run should be disabled (Pr xx.13=0). After the reset the user can reinitialise the timer with a more reasonable wrap-around number.

<b>Pr 85.05</b>	<b>Timer unit timer capture cache</b>		
<b>Access</b>	RW	<b>Range</b>	16 bit
<b>Default</b>	N/A	<b>Update Rate</b>	Immediate

This parameter stores the cached value when using the timer in modes 1 or 2 (Capture modes).

## 6.6 Menu 86 - Digital I/O parameters

The SM-Register has two digital outputs and two digital inputs on-board. These outputs and inputs are controlled through this menu. See Section 3 (Installation) for detailed specifications.

<b>Pr 86.01</b>	<b>Digital input 0</b>		
<b>Access</b>	RO	<b>Range</b>	0/1
<b>Default</b>	N/A	<b>Update Rate</b>	Immediate

<b>Pr 86.02</b>	<b>Digital input 1</b>		
<b>Access</b>	RO	<b>Range</b>	0/1
<b>Default</b>	N/A	<b>Update Rate</b>	Immediate

These two parameters read the state of digital inputs 0 and 1. Inactive input (low) will give the value 0 and active input (high) will give 1.

<b>Pr 86.03</b>	<b>Digital output 0</b>		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	N/A	<b>Update Rate</b>	Immediate

<b>Pr 86.04</b>	<b>Digital output 1</b>		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	N/A	<b>Update Rate</b>	Immediate

Pr **86.03** & Pr **86.04** control digital outputs 0 and 1. Setting to 0 will place the output low and setting to 1 will place the input high (+24V).

<b>Pr 86.05</b>	<b>Digital outputs 0 and 1</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 3
<b>Default</b>	N/A	<b>Update Rate</b>	Immediate

This parameter provides control for both digital outputs and is an alternative to controlling each output individually with Pr **86.03** and Pr **86.04**. Bit0 of this parameter controls digital output 0 (Pr **86.03**) and Bit1 controls digital output 1 (Pr **86.04**).

## 6.7 Menu 88 - Status parameters

Pr 88.01	Error code / reset		
Access	RW	Range	0 to 9999
Default	N/A	Update Rate	On error

This parameter has two purposes - when read it will return the identical run-time error as Pr **81.50** (note - it will not return drive trip codes). The parameter is cleared to zero on reset and when the user program execution is started.

When the parameter is written to with a value of 1070 the SM-Register will initiate a warm-restart of the drive and any other options. This can be used to restart the user program (providing auto-run Pr **81.13**=1) and clear any drive trip. This reset action can be performed at any time, not just after a run-time error or in an ERROR task.



Writing 1070 to Pr **88.01** will result in any drive trip being automatically cleared as well as resetting all installed options in the drive. This behaviour is different to the UD70 product on Unidrive 1 where the drive was not reset.

Pr 88.02	Task in error		
Access	RO	Range	0 to 50
Default	N/A	Update Rate	On Error

The Task in Error parameter can be used to identify which task the error was generated in. This parameter is only valid if it is read from the ERROR task after a run-time trip has occurred. The values will have the following meanings:

Value	Task
50	System
1	Initial
2	Background
3	Clock
4	Error
5	Pos0
6	Pos1
7	Event
8	Event1
9	Event2
10	Event3
11	APC

A value of zero will be returned if there is no error condition. For more information on these parameters refer to Chapter 12 *Diagnostics* on page 139.

Pr 88.03	POS resource monitoring		
Access	RW	Range	0/1
Default	0	Update Rate	Immediate

This parameter allows the user to enable or disable monitoring of the motion engine tasks free resource. When set to 1, Pr **88.04** and Pr **88.05** become active. If set to zero, Pr **88.04** and Pr **88.05** will read zero.

<b>Pr 88.04</b>	<b>Free resource for motion engine tasks</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 95
<b>Default</b>	0	<b>Update Rate</b>	See Pr 81.12

This parameter indicates the percentage resource available for running the motion engine tasks. These tasks are CTSync, CTSync Output Channels, POS0, PLCopen, APC, APC Output Channel and POS1. If this parameter value reaches zero a task overrun will occur. It is calculated every motion engine period and is displayed for the previous motion engine period.

<b>Pr 88.05</b>	<b>Motion engine peak resource detect</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 95
<b>Default</b>	0	<b>Update Rate</b>	See Pr 88.04

This parameter displays the lowest value (i.e. highest resource usage) that Pr 88.04 reaches since the monitoring was enabled (Pr 88.03). It will give a realistic indication of the worst case available resources for the motion engine tasks so that the user can see how close the module has been to a motion engine task overrun.

<b>Pr 88.06</b>	<b>CLOCK task resource monitoring</b>		
<b>Access</b>	RO	<b>Range</b>	0/1
<b>Default</b>	NA	<b>Update Rate</b>	Immediate

This parameter allows the user to enable or disable monitoring of the CLOCK task free resource. When set to 1, Pr 88.07 and Pr 88.08 become active. If set to zero, Pr 88.07 and Pr 88.08 will read zero.

<b>Pr 88.07</b>	<b>Free resource for clock task</b>		
<b>Access</b>	RO	<b>Range</b>	0 to 95
<b>Default</b>	NA	<b>Update Rate</b>	See Pr 81.11

This parameter indicates the percentage resource available for running the Clock task. If this parameter value reaches zero a task overrun will occur. It is calculated every Clock period and is displayed for the previous Clock period.

<b>Pr 88.08</b>	<b>Clock task peak resource detect</b>		
<b>Access</b>	RO	<b>Range</b>	0 to 95
<b>Default</b>	NA	<b>Update Rate</b>	See Pr 81.11

This parameter displays the lowest value (i.e. highest resource usage) that Pr 88.07 reaches since the monitoring was enabled (Pr 88.06). It will give a realistic indication of the worst case available resources for the Clock task so that the user can see how close the module has been to a Clock task overrun.

## 6.8 Menu 90 - General parameters

This menu contains the reference and feedback values from the drive as well as other status information.

**NOTE** When porting UD70 programs to SM-Register special care must be taken as these parameters are different than those on UD70.

Pr 90.01	Feedback encoder position ( $2^{32}/\text{rev}$ )		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	See Pr 81.16

Contains the feedback encoder position.

The top 16-bits are scaled to 65536 counts per rev regardless of the type of feedback device or scaling configured in the drive. The lower 16-bits give the fine position as available from the feedback device scaled to 65536. For standard encoders this will typically be zero, but for higher precision devices such as SinCos encoders, this extra precision will be available.

Marker pulses, etc. have no influence on this parameter.



More information on the use of these feedback parameters can be found in the on-line help of SyPTPro.

Pr 90.02	Feedback encoder revolution count		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	See Pr 81.16

Contains the feedback encoder revolution count.

Pr 90.03	Reference encoder position ( $2^{32}/\text{rev}$ )		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	See Pr 81.16

Contains the reference encoder position.

The top 16-bits are scaled to 65536 counts per rev regardless of the type of feedback device or scaling configured in the drive. The lower 16-bits give the fine position as available from the feedback device scaled to 65536. For standard encoders this will typically be zero, but for higher precision devices such as SinCos encoders, this extra precision will be available.

Marker pulses, etc. have no influence on this parameter.

Pr 90.04	Reference encoder revolution count		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	See Pr 81.16

Contains the reference encoder revolution count.

Pr 90.10	Drive mode		
Access	RO	Range	Signed 16-bit
Default	N/A	Update Rate	Immediate

Provides a definitive method of identifying the mode the drive is in. It is recommended that this parameter is used instead of Pr **11.31** or Pr **00.48** since those parameters indicate the requested, not the actual mode.

The values are defined as follows.

Value	Mode
26	Open-loop
27	Closed-loop vector
28	Servo
29	Regen

In order to programmatically change the drive mode, use the MODEXFER or CMODEXFER function blocks.

Pr 90.11	Drive status and control word		
Access	RW	Range	Signed 16-bit
Default	N/A	Update Rate	Immediate

Writing to this parameter updates the control word. Reading from this parameter reads the status word (same as Pr **10.40**).

**Table 6.5 Control word**

Bit	Description
b15	If set, the value of Pr <b>01.46</b> is set from b6
b14	If set, the value of Pr <b>01.45</b> is set from b5
b13	Sets the value of Pr <b>18.33</b> (Application menu 1, bit 3)
b12	If set, the value of Pr <b>06.32</b> is set from b3
b11	If set, the value of Pr <b>06.31</b> is set from b2
b10	If set, the value of Pr <b>06.30</b> is set from b1
b9	If set, the value of Pr <b>06.15</b> is set from b0
b8	Sets the value of Pr <b>18.32</b> (Application menu 1, bit 2)
b7	Sets the value of Pr <b>18.31</b> (Application menu 1, bit 1)
b6	Sets the value of Pr <b>01.46</b> (Preset select bit 1)
b5	Sets the value of Pr <b>01.45</b> (Preset select bit 0)
b4	User trip. Trips drive immediately if set.
b3	Sets the value of Pr <b>06.32</b> (Sequencing bit 2: Run Reverse)
b2	Sets the value of Pr <b>06.31</b> (Sequencing bit 1: Jog)
b1	Sets the value of Pr <b>06.30</b> (Sequencing bit 0: Run forward)
b0	Sets the value of Pr <b>06.15</b> (Drive enable)

**Table 6.6 Status word**

Bit	Description
b15	Not used
b14	Pr <b>10.15</b> (Mains Loss)
b13	Pr <b>10.14</b> (Direction running)
b12	Pr <b>10.13</b> (Direction commanded)

Table 6.6 Status word

Bit	Description
b11	Pr <b>10.12</b> (Braking resistor alarm)
b10	Pr <b>10.11</b> (Braking IGBT active)
b9	Pr <b>10.10</b> (Regenerating)
b8	Pr <b>10.09</b> (Drive output is at current limit)
b7	Pr <b>10.08</b> (Load reached)
b6	Pr <b>10.07</b> (Above set speed)
b5	Pr <b>10.06</b> (At speed)
b4	Pr <b>10.05</b> (Below set speed)
b3	Pr <b>10.04</b> (Running at or below min speed)
b2	Pr <b>10.03</b> (Zero speed)
b1	Pr <b>10.02</b> (Drive running)
b0	Pr <b>10.01</b> (Drive OK)

Pr 90.12	Event task schedule reason		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	On Event

For description see below.

Pr 90.13	Event1 task schedule reason		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	On Event1

For description see below.

Pr 90.14	Event2 task schedule reason		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	On Event2

For description see below.

Pr 90.15	Event3 task schedule reason		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	On Event3

For description see below.

The four parameters above (Pr **90.12** - Pr **90.15**) give the reason why the particular EVENT task was scheduled. The value only has meaning when the particular EVENT task is running.

The value is bitmapped and defined as follows:

Bits	Description	Value
0-1	Slot triggering the task	0 = Local slot. 1 = Slot 1. 2 = Slot 2. 3 = Slot 3 or Embedded.
2-7	Reason for trigger.	0-31 = Other option module initiated. 32 = CNet Sync. 33 = Timer Unit. 34-63 = User-defined reason via the DPL command SCHEDULEEVENT.

Pr 90.18	Feedback encoder freeze flag		
Access	RW	Range	0/1
Default	N/A	Update Rate	250µs

This parameter needs to be set to zero for the freeze position to be captured. Once the freeze has occurred this parameter is set to 1. To reactivate it simply set it to zero.

Pr 90.19	Feedback encoder freeze position		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	250µs

For description see Pr 90.20.

Pr 90.20	Feedback encoder freeze turns		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	250µs

These 2 parameters store the position and turns of the feedback encoder at the time the freeze input has been activated.

Pr 90.21	Disable drive encoder position check		
Access	RW	Range	0/1
Default	0	Update Rate	Immediate

The drive regularly checks the position derived with the sine and cosine waveforms from a SINCOS encoder via serial communications. Set this parameter to 1 to disable this.

Pr 90.22	Drive encoder comms transmit register		
Access	RW	Range	Unsigned 16-bit
Default	N/A	Update Rate	Immediate

When the Drive Encoder Position Check parameter is disabled (Pr 90.21=1) this parameter can be used to communicate with the encoder connected via serial comms with the drive.

Pr 90.23	Drive encoder comms receive register		
Access	RW	Range	Unsigned 16-bit
Default	N/A	Update Rate	Immediate

When the Drive Encoder Position Check parameter is disabled (Pr 90.21=1) this parameter can be used to communicate with the encoder connected via serial comms with the drive.

Pr 90.24	Module slot number		
Access	RO	Range	Unsigned 8-bit
Default	N/A	Update Rate	Initialisation

This parameter reports the slot number into which the module is installed.

Pr 90.25	Feedback encoder marker position ( $2^{32}/\text{rev}$ )		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	See Pr 81.16

The top 16-bits are scaled to 65536 counts per revolution regardless of the type of feedback device or scaling configured in the drive.



Pr 90.26	Feedback encoder marker turns (2 <sup>16</sup> /rev)		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	See Pr 81.16

This parameter gives the feedback encoder marker revolution count.

Pr 90.27	Application processor database version number		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	Initialisation

The database version number is read from the database after power-up.

Pr 90.28	Reference encoder freeze flag		
Access	RW	Range	0/1
Default	N/A	Update Rate	250µs

This parameter needs to be set to zero for the freeze position to be captured. Once the freeze has occurred this parameter is set to 1. To reactivate it simply set it to zero.

Pr 90.29	Reference encoder freeze position		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	250µs

See Pr 90.30 for description.

Pr 90.30	Reference encoder freeze turns		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	250µs

These 2 parameters store the position and turns respectively of the reference encoder at the time the freeze input was activated.

Pr 90.31	Feedback encoder turns and coarse position		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	See Pr 81.16

See Pr 90.32 for description.

Pr 90.32	Reference encoder turns and coarse position		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	See Pr 81.16

These 2 parameters (Pr 90.31 & Pr 90.32) store the 16-bit turns in the upper word and 16-bit position in the lower word, of the feedback (Pr 90.31) and reference (Pr 90.32) encoders.

Pr 90.33	Feedback encoder freeze turns and coarse position		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	250µs

See Pr 90.34 for description.

Pr 90.34	Reference encoder freeze turns and coarse position		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	250µs

These 2 parameters (Pr 90.33 & Pr 90.34) store the 16-bit turns in the upper word and the 16-bit position in the lower word, at the time the freeze input was activated.

Pr 90.35	Reference encoder marker position ( $2^{32}/rev$ )		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	See Pr 81.16

This parameter stores the reference encoder position at the time the marker pulse was activated.

Pr 90.36	Reference encoder marker turns ( $2^{16}/rev$ )		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	See Pr 81.16

This parameter stores the reference encoder revolution count at the time the marker pulse was activated.

Pr 90.37	Feedback encoder marker turns and coarse position		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	See Pr 81.16

See Pr 90.38 for description.

Pr 90.38	Reference encoder marker turns and coarse position		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	See Pr 81.16

These 2 parameters (Pr 90.37, Pr 90.38) store the 16-bit turns in the upper word and 16-bit position in the lower word, of the feedback (Pr 90.37) and reference (Pr 90.38) encoders at the time the marker pulse was activated.

Pr 90.39	Drive keypad button status		
Access	RO	Range	Signed 16-bit
Default	N/A	Update Rate	> 40ms

The state of the Reverse, Run and Stop keys can be read using this parameter. Button presses from both LCD and LED Keypads will be seen. Keys are represented by bits as follows:

Value	Description	Value	Description
b0	Rev	b2	Stop
b1	Run		

Pr 90.40	Event task trigger		
Access	RW	Range	Unsigned 16-bit
Default	0	Update Rate	Immediate

Upon setting this parameter to a value it will execute one of the SM-Register Event tasks.

Pr 90.41	Reference encoder marker flag		
Access	RW	Range	0/1
Default	N/A	Update Rate	See Pr 81.16

Value	Action	Value	Action
0	Do not trigger Event task	3	Trigger Event2 task
1	Trigger Event task	4	Trigger Event3 task
2	Trigger Event1 task		

See Pr 90.42 for description.

Pr 90.42	Feedback encoder marker flag		
Access	RW	Range	0/1
Default	N/A	Update Rate	See Pr 81.16

These 2 parameters (Pr 90.41 & Pr 90.42) are set to 1 if a relevant encoder marker pulse is activated, but only if the marker flag enable parameters have been set (parameters Pr 90.45 and Pr 90.46). To re-arm the marker these parameters must be set to zero by the user. They cannot be set to 1 by the user.

Pr 90.43	Reference encoder source		
Access	RW	Range	Unsigned 8-bit
Default	N/A	Update Rate	Immediate

See Pr 90.44 for description.

Pr 90.44	Feedback encoder source		
Access	RW	Range	Unsigned 8-bit
Default	N/A	Update Rate	Immediate

These 2 parameters (Pr 90.43 & Pr 90.44) define the source for the reference and feedback data. See the table below for the valid sources.

Value	Description
0	Drive encoder
1	Slot 1
2	Slot 2
3	Slot 3
4	User program
5	Unconfigured

Pr 90.45	Reference marker flag enable		
Access	RW	Range	0/1
Default	N/A	Update Rate	Immediate

See Pr 90.46 for description.

Pr 90.46	Feedback Marker Flag Enable		
Access	RW	Range	0/1
Default	N/A	Update Rate	Immediate

These 2 parameters (Pr 90.45 & Pr 90.46) must be set to 1 to allow the marker flags (Pr 90.41 and Pr 90.42) to be set when the marker pulse is activated.

<b>Pr 90.47</b>	<b>Reference freeze enable</b>		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	N/A	<b>Update Rate</b>	Immediate

<b>Pr 90.48</b>	<b>Feedback freeze enable</b>		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	N/A	<b>Update Rate</b>	Immediate

These 2 parameters (Pr 90.47 & Pr 90.48) must be set to 1 allow the freeze flags (parameters Pr 90.18 and Pr 90.28) to be set when the freeze input is activated.

<b>Pr 90.49</b>	<b>APC runtime error ID</b>		
<b>Access</b>	RO	<b>Range</b>	32bit
<b>Default</b>	N/A	<b>Update Rate</b>	Immediate

This parameter shows the error ID of an APC runtime error. This will be set when module trip 81 occurs. Brief descriptions of the error codes are shown in the following table. For more information, refer to the *Advanced Position Control User Guide*.

<b>Value</b>	<b>Description</b>
0	No error or failure to set up encoder object
1	CAM array too small
2	CAM segment overrun
3	CAM size has been specified as zero
4	CAM absolute reset failure

## 6.9 Menu 91 - Fast access parameters

The parameters in this menu are SM-Register virtual parameters which provide a faster update rate or enhanced resolution than drive parameters.

<b>Pr 91.01</b>	<b>Short-cut enable</b>		
<b>Access</b>	RW	<b>Range</b>	Unsigned 8-bit
<b>Default</b>	0	<b>Update Rate</b>	Immediate

This parameter enables the short-cut parameters detailed later in this section. You must set the appropriate bit in this parameter. See the following table.

<b>Bit</b>	<b>Function</b>	<b>Related Parameter</b>
0	Speed reference shortcut enable	Pr 91.02
1	Hard-speed reference shortcut enable	Pr 91.03
2	Torque reference shortcut enable	Pr 91.04
3	Analog Output 1	Pr 91.11
4	Analog Output 2	Pr 91.12
5	Reserved	-
6	Reserved	-
7	Reserved	-

Pr 91.02	Speed set-point (Pr 01.21)		
Access	RW	Range	Signed 32-bit
Default	N/A	Update Rate	250µs

Sets the speed reference in units of **0.001RPM**. This value is mirrored in drive Pr **01.21** (preset speed 1), therefore in order to control the drive speed with this parameter ensure preset speed 1 is selected on the drive (Pr **01.14**=3, Pr **01.15**=1).

Ensure bit 0 of Pr **91.01** is set and the full-scale speed in Pr **91.05** is set accordingly when using this parameter.

Pr 91.03	Hard-speed reference (Pr 03.22)		
Access	RW	Range	Signed 32-bit
Default	N/A	Update Rate	250µs

Controls the hard-speed reference on the drive in units of **0.001RPM**.

Ensure bit 1 of Pr **91.01** is set and the full-scale speed in Pr **91.05** is set accordingly when using this parameter. Note that this parameter is valid only in Closed Loop Vector and Servo modes only.

Pr 91.04	Torque setpoint (Pr 04.08)		
Access	RW	Range	Signed 32-bit
Default	N/A	Update Rate	250µs

Specifies the torque setpoint (drive Pr **04.08**) in units of 0.01%.

Ensure bit 2 of Pr **91.01** is set in order to use this parameter.

Pr 91.05	Full scale speed (RPM)		
Access	RW	Range	Signed 32-bit
Default	3000	Update Rate	N/A

Set this to the maximum (absolute) speed that will ever be written to with Pr **91.02** or Pr **91.03**. This is in units of 1 RPM.

This determines the resolution for the speed values sent to the drive. Attempting to write speed values to Pr **91.02** or Pr **91.03** greater than the RPM value specified in Pr **91.05** will result in the value being limited or a value overrange run-time error.

Pr 91.06	Speed feedback		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	250µs

This parameter returns the value of the drive speed feedback in units of 0.01RPM in closed loop modes. This parameter will only be updated if the speed feedback is derived from the drive's encoder input, not a slot number. This can only happen if Pr **03.26** on the drive is set to a zero. However, if a low resolution encoder is used there maybe some jitter at low speed. For example, at 10rpm with a 1024ppr encoder this parameter may jump between 0 and 14.65rpm. This is similar to the drive parameter Pr **03.02**.

Pr 91.07	Current feedback (Pr 04.02)		
Access	RO	Range	Signed 16-bit
Default	N/A	Update Rate	250µs

This value is taken from parameter Pr **04.02** and is in 0.01A units (i.e. 150=1.5 Amps).

Pr 91.08	Drive analog input 1 value		
Access	RO	Range	±4000
Default	N/A	Update Rate	250µs

This value will be taken from the drive's analog input 1 and is scaled for ±4000 to represent the +/- full scale signal at the input. Refer to the *Drive User Guide* for information on the sampling rate of analog inputs.

Pr 91.09	Drive analog Input 2 value		
Access	RO	Range	±1023
Default	N/A	Update Rate	250µs

This value will be taken from the drive's analog input 2 and is scaled for ±1023 to represent the +/- full scale signal at the input. Refer to the *Drive User Guide* for information on the sampling rate of analog inputs.

Pr 91.10	Drive analog input 3 value		
Access	RO	Range	±1023
Default	N/A	Update Rate	250µs

This value will be taken from the drive's analog input 3 and is scaled for ±1023 to represent the +/- full scale signal at the input. Refer to the *Drive User Guide* for information on the sampling rate of analog inputs.

Pr 91.11	Drive analog output 1		
Access	RW	Range	±1023
Default	N/A	Update Rate	NA

This parameter sets the value of analog output 1. Refer to Pr **91.01** for information on enabling this parameter. This parameter sets the value of analog output 2. Refer to Pr **91.01** for information on enabling this parameter

Pr 91.12	Drive analog output 2		
Access	RW	Range	±1023
Default	N/A	Update Rate	NA

**NOTE** When the appropriate shortcut is set (Pr **91.01**), the SM-Register can control the drive's analog outputs directly. The values from Pr **91.11** and Pr **91.12** will only be transferred if the drive analog outputs are not in high speed mode. The values are not scaled by the drive and the specified analog output drive mode parameters will not affect the scaling, but the output mode is affected by the analog output drive mode parameters. For example, when a current mode is selected (4-20mA or 0-20mA) on the drive, the output will be in current mode but the scaling will not be affected by which one is selected. In current mode +/-1023 corresponds to +/-21.4mA  
In voltage mode +/-1023 corresponds to +/-10V

<b>Pr 91.16</b>	<b>Drive digital inputs</b>		
<b>Access</b>	RO	<b>Range</b>	Unsigned 8-bit
<b>Default</b>	N/A	<b>Update Rate</b>	250µs

This parameter is similar to drive parameter Pr **08.20** in providing the status of 7 digital inputs in one single parameter. Logic polarity and inversions are taken into account.

The bits are assigned as follows:

Bit	Digital Input
0	F1
1	F2
2	F3
3	F4
4	F5

Bit	Digital Input
5	F6
6	Enable
7	<i>Reserved - Read as zero</i>

<b>Pr 91.17</b>	<b>Number of valid CTSync messages received</b>		
<b>Access</b>	RW	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	

This parameter will increment every time a good CTSync message is received with a valid checksum.

<b>Pr 91.18</b>	<b>Number of bad CTSync messages received</b>		
<b>Access</b>	RW	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	

This parameter will increment every time a CTSync message is received with a bad checksum.

<b>Pr 91.19</b>	<b>Number of missing CTSync messages</b>		
<b>Access</b>	RW	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	

This parameter will increment every time a message has not been received when the module is expecting one.

<b>Pr 91.20</b>	<b>CTSync synchronisation signal width too short</b>		
<b>Access</b>	RW	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	Synchronisation

This parameter will increment every time the synchronisation signal is the incorrect width. During synchronisation this parameter is likely to increase but should stabilise once synchronisation is complete. After synchronisation, if this parameter is incrementing then there is likely to be excessive noise on the EIA-485 network. Check the connections.

<b>Pr 91.21</b>	<b>Inter-option synchronisation control</b>		
<b>Access</b>	RW	<b>Range</b>	0 to 2
<b>Default</b>	0	<b>Update Rate</b>	Immediate

This parameter allows the user to set up the SM-Register in the Inter-Option module Synchronisation scheme. For more information refer to Chapter 11 *Inter-option synchronisation* on page 131.

Bit	Description
0	Set this bit for the SM-Register to participate in the Inter-Option Module Synchronisation scheme as a Producer. Refer to section 11.1 <i>Overview</i> on page 131 for details of the term Producer.
1	Set this bit for the SM-Register to participate in the Inter-Option Module Synchronisation scheme as a Consumer. Refer to section 11.1 <i>Overview</i> on page 131 for details of the term Consumer.

Pr 91.22	Inter-option synchronisation status		
Access	RO	Range	Unsigned 8-bit
Default	N/A	Update Rate	Immediate

This parameter shows the status of the SM-Register in the Inter-option module synchronisation scheme. For more information refer to Chapter 11 *Inter-option synchronisation* on page 131.

Bit	Meaning	Description
0	Requested Inter-Module Synchronisation Role	This is identical to bit 1 of the Inter-Option Module Synchronisation Control Parameter as described above.
1	Requested Inter-Module Synchronisation Role	This is identical to bit 0 of the Inter-Option Module Synchronisation Control Parameter as described above.
2	Inter-Module Synchronisation Role Achieved	This bit indicates that for a module attempting to become inter-module synchronisation Producer it has achieved that role. In the event of more than one module on a given drive attempting to become the synchronisation Producer at least one of these modules will have this bit clear. For a module attempting to become inter-module synchronisation Consumer it indicates that it has not requested to become Producer, and that a Producer has been located in another slot and is being used as the source of synchronisation data. (If a Producer has been located in another slot but the rate of the synchronisation data is not compatible with the Consumer then this bit will be cleared because the Producer - although located - is not being used as the source of synchronisation data).
3	Synchronisation Producer Output within Specification	This bit is relevant only for modules which have been designated as the inter-option module synchronisation Producer and CTSync Slave. This bit indicates that the signal being provided by the inter-option module synchronisation Producer is within the specified tolerance of the drive, and that the drive is now locked to the CTSync Slave's frequency. Note that this bit may only be set if bits 2 to 0 of this parameter are 1,0,1, indicating that the module is Producer (0,1) and that the Producer role has been achieved (1).
7 to 4	Reserved	Read as zero.



# 6.10 Menus 18,19 - Application parameters

These two menus are designated as application parameters since they are all free to be used for whatever purpose the user wants.

Both menus are identical in their layout. All parameters are read/write access to the SM-Register (and via comms), but may be read-only on the drive's keypad.

<b>Pr 1x.01</b>	<b>Integer, read-write saved on power-down</b>		
<b>Access</b>	RW	<b>Range</b>	Signed 16-bit
<b>Default</b>	0	<b>Update Rate</b>	N/A

This parameter is automatically saved by the drive on power-down.

<b>Pr 1x.02-Pr 1x.10</b>	<b>Integer, read-only</b>		
<b>Access</b>	RW (RO drive)	<b>Range</b>	Signed 16-bit
<b>Default</b>	0	<b>Update Rate</b>	N/A

<b>Pr 1x.11-Pr 1x.30</b>	<b>Integer, read-write</b>		
<b>Access</b>	RW	<b>Range</b>	Signed 16-bit
<b>Default</b>	0	<b>Update Rate</b>	N/A

<b>Pr 1x.31-Pr 1x.50</b>	<b>Bit, read-write</b>		
<b>Access</b>	RO	<b>Range</b>	0/1
<b>Default</b>	0	<b>Update Rate</b>	N/A

Pr 1x.11-Pr 1x.50 are saveable in the drive's non-volatile memory.

# 6.11 Menu 20 - Application menu

This menu, like menus 18 and 19, contains parameters that do not affect the operation of the drive and therefore can be used for general purpose.

**NOTE** This menu is NOT saved in the drive's non-volatile memory. Instead it can be stored in the SM-Register flash memory upon request. If more than one SM-Register, SM-Applications or Motion Processor is installed, only one should be configured to store and restore this menu for obvious reasons.

Pr 81.21 is set, this menu will be saved and restored by the SM-Register.

<b>Pr 20.01- Pr 20.20</b>	<b>Integer, read-write</b>		
<b>Access</b>	RW	<b>Range</b>	Signed 16-bit
<b>Default</b>	0	<b>Update Rate</b>	N/A

<b>Pr 20.21 Pr 20.40</b>	<b>Long integer, read-write</b>		
<b>Access</b>	RW	<b>Range</b>	Signed 32-bit
<b>Default</b>	0	<b>Update Rate</b>	N/A

On the drive display/keypad if the value exceeds the maximum that can be displayed (9,999,999), "-----" will be shown. It is not possible to enter values larger than the maximum display value on the keypad.

## Menu 97 - Application processor parameters

Pr 97.00 to Pr 97.99 are 32 bit parameters and are reserved for the Application processor. They can be read and written to from the user program, but are not accessible by the LCD or LED keypads.

This chapter covers:

- Basic DPL program structure and syntax
- Basic DPL commands
- New features offered by the SM-Register

**NOTE** The full reference for all DPL commands and function blocks is provided by the on-line help guides.

7.1 Program header

Every DPL program starts with a header section. SyPTPro creates this section for the user. It basically consists of:

- Program title.
- Program author.
- Program version number.

7.1.1 Aliases

Immediately below the header the user may enter a section of aliases. Aliases are used to 'replace' various expressions or constants:

- a numerical constant expression.
- the address of a register or parameter.
- a DPL expression or statement.

Aliases are created with the \$DEFINE statement.

```
$DEFINE name value
```

For example it is good practice to use aliases to give names to all drive parameters used in a program.

```
$DEFINE PRESET_REF_1 #1.21
$DEFINE PRESET_REF_2 #1.22
$DEFINE SPEED_FB #3.02
```

It is also recommended to have the alias name in UPPER-case letters in order to help distinguish them from normal variables.

**NOTE** It is recommended that aliases representing integer values have a '%' symbol appended to the alias name. In graphical programming tools (QLD/FBD), SyPTPro will treat all aliases without a % symbol as floating-point values. Hence they will be rejected on LD or integer only inputs.

The \$DEFINE directive does NOT produce any code, nor does it speed up the execution time of your program - it simply allows you to refer to something with a different name.

## 7.2 Tasks

A DPL program is separated into sections called tasks. Within the tasks a user writes the program instructions that will be executed by the microprocessor under certain conditions or on a particular time-base. Each task has a particular name, purpose and priority and only one of each task can be present in a DPL program. The common tasks are outlined below:

**Table 7.1 Common tasks**

Task Name	Priority	Purpose
INITIAL	2	The very first task that is run after a power-up or reset. This task is commonly used to initialise drive parameters and program variables. Only the ERROR task can run before this task is completed.
BACKGROUND	6	Low priority task used to do non-time critical functions. This task closely resembles the scan loop of a PLC in the way that it works. It is usual that this task section is created as one big loop, with a command at the end of the task to jump back to the start. If the task is allowed to finish, it will not execute again.
CLOCK	5	Task executed on a fixed timebase (between 1-200ms) used for some time related operations, for example to generate a ramp profile. This task is now synchronised to the drive's level 2 control loop and can be used in place of the old Encoder task.
POS0 POS1	4	Two real-time tasks that run synchronously to a multiple of the drive control loops (range from 250 $\mu$ s to 8ms). These tasks are commonly used to control the drive speed and/or current loop in applications such as positioning. The POS0 task is first to run, followed immediately after by the POS1 task.
EVENT	3	Event tasks only run when a certain event occurs. Events can be raised from various sources such as CTNet, other option modules in the drive or the user program and usually only have a very small number of instructions. They can be likened to interrupt service routines.
EVENT1	3	See above description.
EVENT2	3	See above description.
EVENT3	3	See above description.
ERROR	1	A task that runs only when a run-time error occurs within the user DPL program (for example a divide by zero). This can be used to safely handle abnormal program behaviour situations. All other tasks will be halted prior to the ERROR task running.

**NOTE** When using the CLOCK, POS0 and POS1 tasks, it is advisable to avoid using code such as the FOR and DO WHILE loops. This may cause a DPL over-run error (tr54).

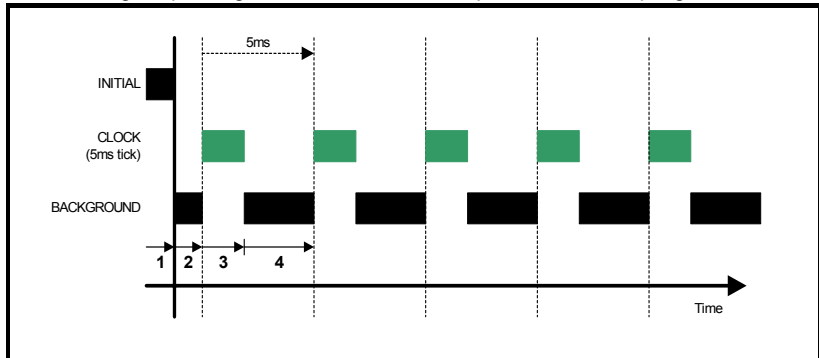
**NOTE** The UD70 ENCODER and SPEED tasks can still be used. These are now aliases for the POS0 and POS1 tasks respectively (i.e. if the program has an ENCODER task this is the same as if it contained a POS0 task). The timebase for both tasks are not fixed as in the UD70 but specified by the user. The CLOCK task in the SM-Register can be used in place of the ENCODER task in the UD70, thus giving a timebase closer to that of the UD70 ENCODER task than is available with the POS0 and POS1 tasks.

All program instructions **must** live within a task. For time-based tasks like POS0, POS1 and CLOCK the instructions within the task have only a finite time in which to complete, therefore only time critical functions should be performed within them.

The position tasks consist of POS0, APC and POS1 and are executed in this order, if applicable (i.e. if they are set to run).

Tasks have different priority levels, therefore it is possible for one task to interrupt another task. In the above table, the higher the priority number the higher the priority is. Therefore a POS0 task can interrupt a CLOCK task which can interrupt the BACKGROUND task.

The following simple diagram illustrates the concept of tasks interrupting each other:

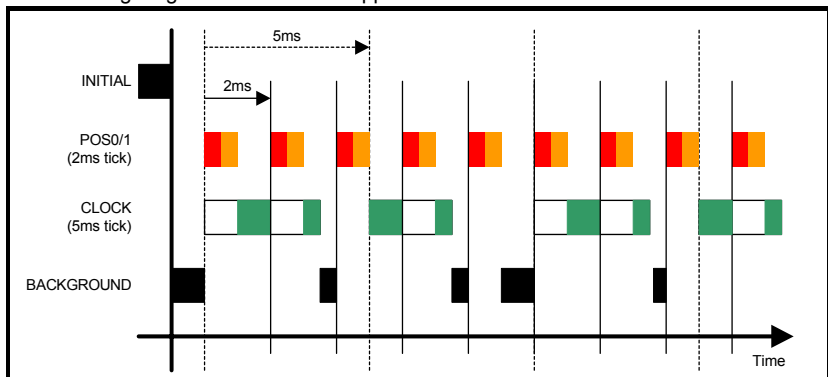


Key:

1. INITIAL task has exclusive control. No other tasks can run.
2. BACKGROUND task runs after INITIAL task has completed.
3. CLOCK task interrupts BACKGROUND task. The drive controls when the CLOCK task will execute. BACKGROUND task is suspended.
4. CLOCK task has finished and now the BACKGROUND task can continue running - until the next clock tick occurs.

Take particular note that the CLOCK task is run on a fixed timebase (in the diagram above it is 5ms). This means that the instructions within the CLOCK task MUST take less than 5ms to complete otherwise the BACKGROUND task will not be able to execute, or a processor overload trip will occur.

The following diagram shows what happens when the POS tasks are set to run as well:



This shows the POS0 and POS1 tasks interrupting the CLOCK task which in turn interrupts the BACKGROUND task. As can be seen, this is quite a heavily loaded program since the background task is only executed once in a while. The processor free resource parameter Pr **81.04** can be used to determine how heavily loaded the SRegister is.

### 7.2.1 EVENT tasks

There are four event tasks provided. The event tasks can be triggered on:

- CTNet SYNC frame received (configured via Pr **81.35**)
  - User program initiated
- New DPL command SCHEDULEEVENT. (See on-line help for information).

## 7.3 Variables

### 7.3.1 Types

There are three basic types of variables:

1. Integer Variable
2. Double-precision Floating Point Variable
3. Single-precision Floating Point Variables

An Integer variable is denoted by a % symbol after the variable name. A Floating Point variable is denoted by the lack of a % symbol.

**Table 7.2 Variable types**

Type	Representation	Range
Integer	32-bit signed.	-2147483648 to 2147483647
Single float	32-bit, 1 sign bit, 8 exponent and 23 mantissa.	$\pm 3.40282e+038$
Double float	64-bits: 1 sign bit, 52 bit mantissa, 11 bit exponent	$\pm 1.79769e+308$

Example of variables:

```
Speed% = 1234    // a integer variable
Value = 55.6     // a floating point variable
```

A special statement is placed at the start of the program to declare what type of floating point variable is used throughout the program - either single or double precision. By default double-precision variables will be used. By including the following line immediately below the program header region (with \$TITLE, etc.) the float type will be single-precision:

```
$flt single
```

### 7.3.2 Variable names

The first character of a variable name must be a letter. Subsequent characters may include letters, numbers and the underscore (\_) character.

#### NOTE

- Variable names are case sensitive (e.g. The variable names speed%, SPEED% and Speed% are different variables).
- SyPTPro QuickLD and FBD editors will only allow the use of variables no longer than 16 characters including any % sign.

### 7.3.3 Initialisation of variables

All variables must be given an initial value before they can be used. This is typically done within the INITIAL task. For example,

```
Initial {
Speed_SP% = 0
Ramp% = 0
}
```

### 7.3.4 Scope and lifetime of variables

Variables can either be global or local. All variables declared in DPL programs are global. i.e. they can be accessed and altered by any task, with the exception of variables within a User Defined Function Block which are local (i.e. cannot be accessed from outside the user defined function block).

No DPL variables survive a reset of the SM-Register. Remember that resetting the drive from a tripped condition will also cause a reset (depending on the value in Pr **xx.15**).

### 7.3.5 Fixed-size variable arrays

A DPL program may contain arrays of either integer or floating-point variables. Only fixed-size (single-dimension) arrays are allowed.

An array must first be declared using the DIM statement (usually in the Initial task), and the number of elements given in square brackets after the variable name, e.g:

```
DIM MyArray[20]    // Integer array having 20 elements
DIM Array2[30]     // Floating point array having 30 elements
```

The elements in an array are numbered 0 to number\_of\_elements - 1. So from the above example, the first element of myarray[] is:

```
myarray[0]
```

and the last is:

```
myarray[19]
```

Two functions are provided that can be used at run-time to determine the upper and lower bounds of an array. These are UPPER and LOWER. for myarray[], UPPER will return 19 and LOWER will return 0.

### 7.3.6 Constant arrays

Constant arrays, as the name suggests, contain fixed pre-defined values. The values of the constant array are defined within the DPL program by using a special section (see CONST in the on-line help). Only integer values can be defined.

The advantage of constant arrays is that the size of the array is only limited by the amount of available program space - and not variable RAM. The program space is 384kb - it is used to store the compiled DPL file, constant array data, and optionally, the DPL file itself.

### 7.3.7 Storage space - number of variables

All variables, fixed-size variable arrays and PLC registers live in an 80kbytes memory space. Each integer variable and single-precision floating point variable consumes 4-bytes (32-bit), and double-precision floating point variables consume 8-bytes (64-bit). There are other things that consume memory as well, such as parameter accesses.

The DPL compiler will notify you if you reach the limit of available memory.

### 7.3.8 Bit addressing of variables

All integer variables and arrays may be bit-addressed. This means that each individual bit within the variable may be separately read or written. After the variable name, place a decimal point (.) followed by the bit number between 0 and 31.

Example 1 (simple variable):

```
Flags% = 0    // initialise all 32 bits to 0
Flags%.0 = 1  // set bit 0 to 1

// now test to see if bit 0 AND bit 1 are set to 1.
IF Flags%.0 & Flags%.1 = 1 THEN
    PRINT "Test satisfied."
ENDIF
```

Example 2 (arrays):

```
DIM MyArray%[10]
...
IF MyArray%.1[4] = 1 THEN;test bit 1 of element #4.
    PRINT "Test satisfied."
ENDIF
```

Note: The bit number must be a constant number - variables are not allowed.

### 7.3.9 PLC registers

The 'PLC' area is a special range of pre-defined 32-bit registers. The PLC registers are split into 6 sets of 100 parameters numbered 00 to 99. The registers can also be accessed from within a user DPL program by a special variable name or array name. Four of the register sets are also saveable in the SM-Register *Flash* memory. See section 6.4 *Menus 70-75 - PLC registers* on page 79 for further information on PLC registers.

#### 7.3.10 RAM files

RAM files enable the user to store 'files' in the user RAM of the SM-Register. These can be uploaded and downloaded using DPL commands. They have an advantage in that you can retrieve or write an array of numbers in one go rather than each element of the array individually.



For further information on RAM files, including example programs, please refer to the on-line help.

## 7.4 Parameters

Parameters are categorised into two sets:

- Drive Parameters
- SM-Register Parameters

Drive parameters are ones which reside in the host drive. The majority of them affect the operation of the drive, however a few are set aside as "application parameters". These are menus 18, 19 and 20.



The SM-Register parameters are local and accessible only to the SM-Register. These parameters provide access to additional features of the SM-Register, and give faster access to some drive parameters.

**NOTE** The SM-Register always guarantees that the drive parameter database it uses matches that of the host Unidrive SP. When a SM-Register is installed to a Unidrive SP for the first time and powered up the word “Loading” may appear on the drive display for a few seconds. This indicates the SM-Register is synchronising databases. This will only occur the first time the module is installed to the drive. Subsequent power-ups may show “Loading” only for a very short time.

7.4.1 Reading and writing parameters

Reading and writing parameters is achieved by using the # command. Parameters are accessed in a Pr **xx.XX** format in the same way as they are accessed on the drive's keypad.

For example, to read the speed feedback parameter (Pr **03.02**), use:

```
Speed% = #3.02
```

to write to a speed reference parameter (eg. Pr **01.22**), use:

```
#01.22 = 1500
```

Note that the leading zero in the menu/parameter field is optional. For example #3.02, #03.02, #03.2 and #3.2 will access exactly the same parameter.

7.4.2 Fixed-point integer parameters

Dealing with fixed-point integer parameters can be quite a bit slower than for integer parameters. In order to speed this up, a special command #INT can be used when reading and writing parameters. When using this command with fixed-point integer the decimal places will be automatically removed.

For example, parameter Pr **1.19** has a range of 0.000 - 0.099. Reading the parameter using:

```
Speed_Fine% = #INT1.19
```

will return integer values between 0 and 99. When writing, the command:

```
#INT1.19 = 45
```

will set the parameter to 0.045 (same as #1.19=0.045). The benefit of this is that the DPL program can use integer variables (%) instead of floating-point, thus providing a speed advantage.

7.5 Operators

DPL offers all the standard operators as follows:

**Table 7.3 Standard operators in precedence order**

Operator	Meaning
-	Arithmetic negation.
!	Boolean negation (unary).
!(..., nbit)	Negation of <i>nbit</i> bits.
*	Multiplication.
/	Division.
%	Modulo (remainder).
+	Addition.

**Table 7.3 Standard operators in precedence order**

Operator	Meaning
-	Subtraction
&	Bitwise AND
	Bitwise OR
^	Bitwise exclusive OR (XOR)

**Table 7.4 Conditional operators in precedence order**

Operator	Meaning
=	Equality
<	Less than
>	Greater than
<=	Less then or equals
>=	Greater than or equals
<>	Inequality
AND	Boolean AND
OR	Boolean OR
NOT	Boolean NOT

## 7.6 Basic DPL commands



Refer to the on-line help for full reference on the DPL language and function block library.

### FOR loop

```
FOR variable = integer_expression to integer_expression [STEP constant]
    statements
LOOP
```

### CASE

This provides an alternative to the IF-ELSEIF-ENDIF construct.

```
SELECT integer_expression
CASE integer_constant
    statements
    [CASE integer_constant, integer_constant ...
        [statements]]
    [ELSE
        [statements]]
ENDSELECT
```

This construct provides a convenient way to test for multiple constant values. Any number of CASE statements can be included.

**NOTE** In the two examples shown above some sections are within square brackets ([ and ])). This section of code within the square brackets is optional.

**NOTE** The CASE statements operate in the same way as programs like Visual Basic in that the program flow will NOT drop through to the next CASE as it does in the C programming language.

### MAX\_INT, MIN\_INT, MIN\_FLOAT, MAX\_FLOAT

These are special predefined keywords that are recognised by the DPL compiler and replaced by the appropriate numeric value.

**Table 7.5 Min/Max**

Keyword	Value
MIN_INT	-2147483648
MAX_INT	2147483647
MIN_FLOAT	-3.40282e+038 (single precision model) -1.79769e+308 (double precision model)
MAX_FLOAT	3.40282e+038 (single precision model) 1.79769e+308 (double precision model)

### UPPER/LOWER

These functions will take an array as a parameter and will return the upper and lower array index respectively. For example:

```
// Create an array of 1000 elements
DIM Array%[1000]

// now,

l% = LOWER(Array%)    // will return the value 0
u% = UPPER(Array%)    // will return the value 999.

// get the sum of all values in array%
Total%=0
FOR i% = LOWER(Array%) TO UPPER(Array%)
    Total% = Total% + Array%[i%] //add array element value to total
LOOP
```

### TRUNC

This is used to convert a floating point value to integer, truncating rather than rounding. For example:

```
// Initialise floating point variable
FloatVal = 1.56

Int1% = FloatVal        // auto-cast rounds to 2.
Int2% = INT(FloatVal)   // explicit cast with INT rounds to 2
Int3% = TRUNC(FloatVal) // explicit cast with TRUNC gives 1
```

Safety Information
Introduction
Installation
Getting started
SM-Register Functionality
Parameters
<b>DPL Programming</b>
Communications
Freeze and Marker
CTSync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

### SCHEDULEEVENT

This function block is used to schedule an EVENT task. The arguments are:

- Slot number  
Specifies which slot to schedule the event task in. Currently only 0 is allowed here which means the local slot.
- Task ID  
Range 0-3 to specify which EVENT task to trigger
- Reason  
A user defined reason. Must be a value of 34 or higher. This value can be accessed in the EVENT task by looking at parameter Pr **90.12**-Pr **90.15**.

```
BACKGROUND {
  ... some code
  // schedule local event1 task with reason code of 45.
  a% = SCHEDULEEVENT(0, 1, 45)
  ... some more code
}

EVENT1 {
  IF #90.13 = 45 THEN
    // task scheduled from DPL
  ENDIF
}
```

### CTNETDIAGNOSTICS

This returns diagnostic information for CTNet. Refer to on-line help. This replaces the special variables such as NOFMESSAGES that was used on the UD70 product.

This command takes no input and returns 10 outputs. The table below details the outputs and the equivalent UD70 variable.

**Table 7.6 CTNetDiagnostics output values**

Output	Value	UD70 Variable
1	Total number of messages handled by this node	NOFMESSAGES
2	Number of cyclic data overruns	NOFOVERRUNS
3	Lost RX messages	NOFLOSTMESSAGES
4	Number of retries	NOFRETRIES
5	Number of recons	NOFRECONS
6	Number of excessive NAKs	NOFEXENAKS
7	Duplicate sync	NOFDUPSYNCS
8	Number of locally generated recons	NOFMYRECONS
9	Number of non-cyclic messages	NOFNONCYCLICMESSAGES
10	Number of lost routing messages	N/A

### GETPARATTR

This is used to get parameter attributes such as maximum and minimum values, read-only flag, etc.

```
(Max%, Min%, Flags%) = GETPARATTR(Menu%, Par%)
```

**CModeXfer**

This allows the user to change the drive mode without any installed option modules getting a hard reset. It allows for a smoother drive mode change. While the drive mode change occurs fieldbuses will not be able to write to parameters and this is handled at system level. They will NOT get a ‘write to parameter failed’ message during this period.

**NOTE** Menus 15 to 20 will not be defaulted when using this command.

**RtuReadRegs**  
**RtuReadParas**  
**RtuReadInputRegs**  
**RtuPresetRegs**  
**RtuPresetParas**  
**RtuMasterReply**  
**RtuMasterStatus**

These commands are implemented to allow the user to utilise the SM-Registers Modbus RTU master functionality. Refer to the on-line help for further details.

**PFIXREAD6/PFIXWRITE6**

These blocks provide reading and writing of drive parameters in a fixed precision of 6 decimal places.

**SETUSERID**

This command is used to set the User ID Pr **81.49**.

```
SETUSERID(101)    // set #81.49 to 101.
```

**ANSIREPLY**

It is a EIA-485 port function which is used in conjunction with the ANSIREAD and ANSIWRITE functions.

ANSIREPLY examples:

```
Result% = ANSIREADN(12, 1811) //perform ansi read

//has message been sent successfully
IF Result% = 0 THEN
    //message not sent successfully
    goto top:
ENDIF

Timeout% = 0 //message sent successfully so initialise timeout
DO
    (status%, reply%) = ANSIREPLY()// get status and value of read
    DELAY(1)           // delay 100ms
LOOP WHILE Status% = -65536 AND Timeout% < 50 //Timeout=50xclock timebase
```

```

Background{

top:

Value% = #18.11
Result% = ANSIWRITEN(12, 1811, value%, 1);write value to remote drive
IF Result% = 0 THEN
    //message not sent successfully
    goto top:
ENDIF

CALL get_reply: //get reply

GOTO top:
} //Background

get_reply:{
Timeout% = 0
DO
    (Status%, Reply%) = ANSIREPLY()
LOOP WHILE Status% = -65536 AND Timeout% < 50
} //get_reply:

```

The first output argument returns the status of the ANSIREPLY command and can be one of the following values:

- 65536 = No reply received yet
- 65537 = Reply received, but with bad checksum
- 65538 = EOT received (i.e. parameter does not exist)
- 65539 = NAK received
- 65540 = ACK received

#### **AssRAM**

#### **UnassRAM**

#### **RamLength**

#### **SetRamLength**

These commands allow the programmer to use the RAM files within the SM-Register. RAM files provide a means of accessing user program arrays via the CMP file services. For more information on these commands and RAM files refer to the on-line help.

#### **SMARTCARD functions**

Functions are available to utilise the SMARTCARD for reading and storage. Refer to the SyPTPro help for more information on using these functions.

### **7.6.1 DPL Commands and function blocks**

There is a rich list of commands and functions that may be used in a DPL program. Please refer to the on-line help.

## 7.7 User defined function blocks

### 7.7.1 Overview

SyPTPro comes as standard with a pre-defined library of function blocks that can be used in the graphical programming tools (LD and FBD) as well as in raw DPL.

The User Defined Function Block system allows the user to create their own function blocks that will automatically become available in the graphical programming tools (Function Block Diagrams and QuickLD diagrams) in addition to the standard library functions.

A UDFB itself is like a self-contained DPL program section in its own right and therefore can consist of a mixture of raw DPL commands, FBD and QLD diagrams and other UDFBs. Note however that you cannot create standard task sections (such as POS0) with a UDFB.

### 7.7.2 Scope of a UDFB

Each UDFB is local to the node's DPL program in which it is defined. To make a UDFB available in other node programs, it is simply a matter of copying and pasting the UDFB section into the other node program.

A UDFB appears within the DPL Editor of SyPTPro in a similar manner to a task - i.e. a collapsible section - and it is recommended practice to place all UDFBs at the top of a program due to the fact that a UDFB must be defined before it is used.

### 7.7.3 Encapsulation and data storage

Unlike any task of a DPL program, UDFBs are self-contained units (i.e. encapsulated). This means that each UDFB has its own unique set of variables (local variables).

A UDFB interfaces to the node's DPL program though its input and output arguments alone. It is not possible for a UDFB to access the global DPL variables in the DPL program, or variables in other UDFBs.

A UDFB can of course access drive parameters and SM-Register registers that are considered global, however this is to be discouraged especially for blocks that could be re-used in other programs or applications. The only times where a block may need to access parameters or registers directly would be in application / product specific situations.

Each time a UDFB is used in a DPL program, a separate instance is made that is a copy of the UDFB with unique local variables.

**NOTE** The local variables of a UDFB cannot be watched in the SyPTPro Watch Window.

### 7.7.4 UDFB naming

In order to keep UDFBs unique and to avoid any naming collisions between UDFBs and the standard library function blocks, a UDFB name must start with the underscore character (\_). The name is also limited to 16 characters, however it is recommended for the name be kept short so that it displays neatly within the SyPTPro FBD and QuickLD editors, e.g.

`_MyFunc`, `_PID1` and `_My_Func`

These are examples of illegal names:

`MyFunc`, `UDFB1`

### 7.7.5 Input and output arguments

A UDFB can have the following data types passed into and out of it:

- Integer variables
- Floating point variables
- Integer arrays
- Floating point arrays

The input and output arguments are standard DPL variables - i.e. case-sensitive and must start with a letter not a number. The length of input argument names is not limited, however the FBD and QuickLD editors within SyPTPro will only show the first 5 characters of the argument.

The quantity of inputs and outputs is limited only by available memory unlike the UD70 product which was limited to 10 integer inputs and 10 integer outputs.

### 7.7.6 UDFB code sections

The code within a UDFB is split into two sections:

- The initial code section
- The body code section

The initial section is used for declaring and initialising any local variables that the UDFB will use. The initial section is run for every instance of a UDFB at start-up or reset (this occurs prior to the DPL Initial task).

**NOTE** The input and output arguments of a UDFB cannot be used in the initial section of the UDFB.

The body section is where the actual code of the function block exists the part that does the work of the function. Input and output arguments only have context within the body section.

The two sections are separated by the keyword FBbody. Initial code goes before this keyword, body code after.

Remember that the actual code can consist of a mixture of DPL, FBD diagrams and QLD diagrams.

Below is an example of a simple UDFB that adds two numbers and scales by a pre-defined amount (0.5):

```
(Output%) = _simplefb(Input1%, Input2%) {  
  // initialisation code:  
  
  Scale% = 500      // initialise a local variable  
  
  FBbody  
  // main body code:  
  
  Output% = Input1% + Input2% * Scale% / 1000  
}
```



# 8 Communications

## 8.1 EIA-485 Serial communications port

The SM-Register comes with a EIA-485 serial communications port on-board. Refer to section 3 *Installation* on page 11 for information on the hardware connections and wiring.

This port supports a number of built-in protocols: CT-ANSI slave, Modbus RTU in master and slave modes, Modbus ASCII in master and slave modes and 3 user modes. Both two and four wire configurations are possible

If an invalid or unsupported mode is selected, the mode will be defaulted back to 1 (4-wire CT-ANSI) and a run-time error 49 may occur.

The baud-rate is specified in Pr **81.07**.

The address of this unit is specified in Pr **81.05**.

### 8.1.1 CT-ANSI

The SM-Register supports the ANSIx3.28 protocol.

**NOTE** The CT-ANSI protocol supports reading and writing of full 32-bit values.

All drive parameters as well as SM-Register parameters can be accessed via the EIA-485 port on the module.

#### 8.1.1.1 Reading a parameter

The following tables show the message constructs for reading a parameter.

**Table 8.1 Master request**

Character	Description
EOT	End of transmission (Ctrl & D).
A1	Application Processor address: 1 <sup>st</sup> digit.
A1	Application Processor address: 1 <sup>st</sup> digit.
A2	Application Processor address: 2 <sup>nd</sup> digit.
A2	Application Processor address: 2 <sup>nd</sup> digit.
M1	Menu number: 1 <sup>st</sup> digit.
M2	Menu number: 2 <sup>nd</sup> digit.
P1	Parameter number: 1 <sup>st</sup> digit.
P2	Parameter number: 2 <sup>nd</sup> digit.
ENQ	Enquiry (Ctrl & E).

**Table 8.2 Slave response (if request was correct and parameter exists)**

Character	Description
STX	Start of text (Ctrl & B).
M1	Menu number: 1 <sup>st</sup> digit.
M2	Menu number: 2 <sup>nd</sup> digit.
P1	Parameter number: 1 <sup>st</sup> digit.
P2	Parameter number: 2 <sup>nd</sup> digit.
D1	Data: 1 <sup>st</sup> digit.

**Table 8.2 Slave response (if request was correct and parameter exists)**

Character	Description
D2	Data: 2 <sup>nd</sup> digit.
-	
-	
Dn	Data: n <sup>th</sup> digit.
ETX	End of text (Ctrl & C).
	Checksum.

If the parameter to be read does not exist, the end of transmission character (Ctrl & D) is returned.

The checksum is derived by exclusive ORing the message bytes (characters) together excluding the STX character and the checksum, i.e.  $\text{Checksum} = M1 \wedge M2 \wedge P1 \wedge P2 \wedge D1 \wedge D2 \wedge \dots \wedge Dn \wedge \text{ETX}$ . The checksum is an unsigned 8bit value and if this value is less than 32 then 32 is added to it.

### 8.1.1.2 Writing to a parameter

The following tables show the message constructs for writing to a parameter.

**Table 8.3 Master request**

Character	Description
EOT	End of transmission (Ctrl & D).
A1	Application Processor address: 1 <sup>st</sup> digit.
A1	Application Processor address: 1 <sup>st</sup> digit.
A2	Application Processor address: 2 <sup>nd</sup> digit.
A2	Second Processor address: 2 <sup>nd</sup> digit.
STX	Start of text (Ctrl & B).
M1	Menu number: 1 <sup>st</sup> digit.
M2	Menu number: 2 <sup>nd</sup> digit.
P1	Parameter number: 1 <sup>st</sup> digit.
P2	Parameter number: 2 <sup>nd</sup> digit.
D1	Data: 1st digit.
D2	Data: 2nd digit.
-	
-	
Dn	Data: nth digit.
ETX	End of text (Ctrl & C).
	Checksum.

The following rules apply to the data field:

1. The maximum length is 12 characters.
2. The field may contain leading spaces, but not after any other character.
3. A sign character is optional. No sign indicates positive.
4. A decimal point is optional. This can appear at any point in the data field, but not before the sign or before 10 numbers (i.e. the value written should not have more than 9 decimal places). If the decimal point is not in the same position as used by the parameter, some accuracy may be lost or extra decimal places added (e.g. if +1.2345 is written to a parameter with one decimal place the result is 1.2, if +1.2 is

written to a parameter with three decimal places the result is 1.200). It should be noted that parameters can only have 0, 1, 2, 3, 4, 5, or 6 decimal places.

5. The data field can contain up to 10 numbers, but the value (even ignoring decimal points) must not exceed the range  $-2^{31}$  to  $2^{31}-1$ .

If the parameter is written successfully an Acknowledge character (Ctrl & F) is returned. If the parameter does not exist, the value written exceeds the range for that parameter or the data field rules are not obeyed, a Not acknowledge character (Ctrl & U) is returned.

The checksum is derived by exclusive ORing the message bytes (characters) together excluding the STX character and the checksum, i.e.  $\text{Checksum} = M1 \wedge M2 \wedge P1 \wedge P2 \wedge D1 \wedge D2 \wedge \dots \wedge Dn \wedge \text{ETX}$ . The checksum is an unsigned 8bit value and if this value is less than 32 then 32 is added to it.

### 8.1.1.3 SM-Register address

The SM-Register will only act on messages received where the full address matches the address of the SM-Register or the group address in the message (1<sup>st</sup> digit) matches the 1<sup>st</sup> digit of the address of SM-Register or the address in the message is a broadcast (0). Broadcast messages are used to write data to multiple nodes.

### 8.1.1.4 Control characters

**Table 8.4 Summary of control characters**

Character	Description	ASCII code	Ctrl code
STX	Start of text	02	B
ETX	End of text	03	C
EOT	End of transmission	04	D
ENQ	Enquiry	05	E
ACK	Acknowledge	06	F
BS	Backspace	08	H
NAK	Not acknowledge	15	U

### 8.1.2 Modbus-RTU

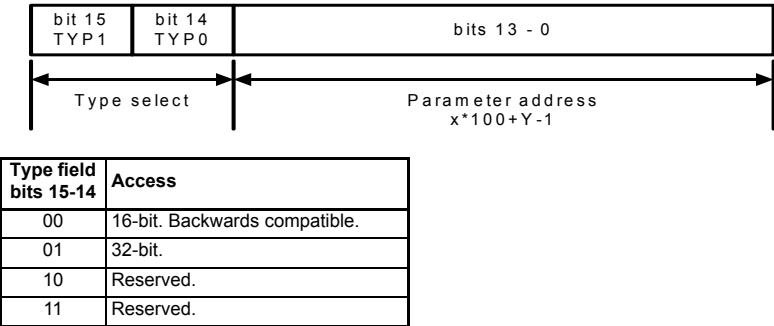
Both slave and master modes of the Modicon Modbus-RTU protocol are supported. In RTU slave mode, the following function codes are supported:

Function	Description
FC3	Read multiple registers.
FC6	Preset single registers.
FC16	Preset multiple registers.
FC23	Read/write multiple registers.

The maximum number of registers that can be read/written simultaneously is 20.

Drive parameters are mapped to Modbus registers as **40000 + (Menu×100) + Parameter**. For example, Pr **01.21** will be register number 40121.

Parameter data can be accessed in either 16-bit or 32-bit mode. The mode is selected using the upper 2 bits of the register address, as follows:



Therefore, to access Pr **70.01** in 32-bit mode, the register number will be 40000 + (0x4000 + 70×100 + 01) = 63385.

If a 32-bit parameter is read with 16-bit access mode, the least significant 16-bits will be returned.

NOTE

The actual register number transmitted at the protocol level is one less than the one asked for and does not include the 40000 offset. Most Modbus masters handle this -1 difference automatically, but some don't.

For master mode the following commands are used in the user DPL program:

- RtuReadHoldingRegs
- RtuReadHoldingParas
- RtuReadInputRegs
- RtuPresetMultipleRegs
- RtuPresetMultipleParas
- RtuMasterReply
- RtuMasterStatus

### 8.1.2.1 FC03 Read multiple registers

Read a contiguous array of 16bit registers. The slave imposes an upper limit on the number of registers which can be read. If this is exceeded the slave will issue an exception code 2.

The following tables show the message constructs for Modbus RTU Function Code 03.

**Table 8.5 Master request**

Byte	Description
0	Slave destination node address 1 through 247, 0 is broadcast.
1	Function code 0x03.
2	Start register address MSB.
3	Start register address LSB.
4	Number of 16bit registers MSB.
5	Number of 16bit registers LSB.
6	CRC LSB.
7	CRC MSB.

**Table 8.6 Slave response**

Byte	Description
0	Slave source node address.
1	Function code 0x03.
2	Length of register data in read block (in bytes).
3	Register data 0 MSB.
4	Register data 0 LSB.
3 + byte count	CRC LSB.
4 + byte count	CRC MSB.

**8.1.2.2 FC06 Preset single register**

Writes a value to a single 16bit register. The normal response is an echo of the request, returned after the register contents have been written. The register address can correspond to a 32bit parameter but only 16bits of data can be sent.  
The following tables show the message constructs for Modbus RTU Function Code 06.

**Table 8.7 Master request**

Byte	Description
0	Slave destination node address 1 through 247, 0 is broadcast.
1	Function code 0x06.
2	Register address MSB.
3	Register address LSB.
4	Register data MSB.
5	Register data LSB.
6	CRC LSB.
7	CRC MSB.

**Table 8.8 Slave response**

Byte	Description
0	Slave source node address.
1	Function code 0x06.
2	Register address MSB.
3	Register address LSB.
4	Register data MSB.
5	Register data LSB.
6	CRC LSB.
7	CRC MSB.

**8.1.2.3 FC16 Preset multiple registers**

Writes a contiguous array of registers. The slave imposes an upper limit on the number of registers which can be written. If this is exceeded the slave will discard the request and the master will timeout.  
The following tables show the message constructs for Modbus RTU Function Code 16.

**Table 8.9 Master request**

Byte	Description
0	Slave destination node address 1 through 247, 0 is broadcast.
1	Function code 0x10.
2	Start register address MSB.
3	Start register address LSB.
4	Number of 16bit registers MSB.
5	Number of 16bit registers LSB.
6	Length of register data to write (in bytes).
7	Register data 0 MSB.
8	Register data 0 LSB.
7 + byte count	CRC LSB.
8 + byte count	CRC MSB.

**Table 8.10 Slave response**

Byte	Description
0	Slave source node address.
1	Function code 0x10.
2	Start register address MSB.
3	Start register address LSB.
4	Number of 16bit registers written MSB.
5	Number of 16bit registers written LSB.
6	CRC LSB.
7	CRC MSB.

#### 8.1.2.4 FC23 Read/Presets multiple registers

Writes and reads two continuous arrays of registers. The slave imposes an upper limit on the number of registers which can be written. If this is exceeded the slave will discard the request and the master will timeout.

The following tables show the message constructs for Modbus RTU Function Code 23.

**Table 8.11 Master request**

Byte	Description
0	Slave destination node address 1 through 247, 0 is broadcast.
1	Function code 0x17.
2	Start register address to read MSB.
3	Start register address to read LSB.
4	Number of 16bit registers to read MSB.
5	Number of 16bit registers to read LSB.
6	Start register address to write MSB.
7	Start register address to write LSB.
8	Number of 16bit registers to write MSB.
9	Number of 16bit registers to write LSB.
10	Length of register data to write (in bytes).
11	Register data 0 MSB.

**Table 8.11 Master request**

Byte	Description
12	Register data 0 LSB.
11 + byte count	CRC LSB.
12 + byte count	CRC MSB.

**Table 8.12 Slave response**

Byte	Description
0	Slave source node address.
1	Function code 0x17.
2	Length of register data in read block (in bytes).
3	Register data 0 MSB.
4	Register data 0 LSB.
3 + byte count	CRC LSB.
4 + byte count	CRC MSB.

### 8.1.3 Modbus ASCII

Both slave and master modes of the Modicon Modbus ASCII protocol are supported. In Modbus ASCII slave mode, the following function codes are supported.:

Function	Description
FC3	Read multiple registers.
FC6	Preset single registers.
FC16	Preset multiple registers.
FC23	Read/write multiple registers.



Refer to on-line help for further information.  
Full details of the Modbus protocol can be found on the Modicon web site at [www.modicon.com](http://www.modicon.com). Note that the 32-bit access mode is specific to Control Techniques.

### 8.1.4 User Modes

These modes turn off all internal protocols and allow the user to access the EIA-485 port directly from the DPL program. They can be used in conjunction with the DPL ANSI commands - ANSIREAD, ANSIWRITE etc. User defined protocols can also be implemented using the DPL PUTCHAR and GETCHAR commands.

## 8.2 CTNet

Full details of CTNet are outside the scope of this User Guide and can be found in the separate CTNet *User Guide*.

## 8.3 SM-Register mapping parameters (fieldbus)

The SM-Register has internal parameters that can be written to or read from by fieldbus options also installed to the drive. This can provide a convenient way to communicate between 2 fieldbuses. These parameters are shown in the table below.

**Table 8.13 SM-Register Internal Parameters**

Second Processor Parameters	Parameter Reference	Direct to Slot 1	Direct to Slot 2	Direct to Slot 3 (Digitax Embedded)
_Pxx% PLC Registers	Pr 70.XX	Pr 100.XX	Pr 130.XX	Pr 160.XX
_Qxx% PLC Registers	Pr 71.XX	Pr 101.XX	Pr 131.XX	Pr 161.XX
_Rxx% PLC Registers	Pr 72.XX	Pr 102.XX	Pr 132.XX	Pr 162.XX
_Sxx% PLC Registers	Pr 73.XX	Pr 103.XX	Pr 133.XX	Pr 163.XX
_Txx% PLC Registers	Pr 74.XX	Pr 104.XX	Pr 134.XX	Pr 164.XX
_Uxx% PLC Registers	Pr 75.XX	Pr 105.XX	Pr 135.XX	Pr 165.XX
Local Configuration Parameters	Pr 81.XX	Pr 111.XX	Pr 141.XX	Pr 171.XX
Timer Function Parameters (Apps, Apps Plus & ST Plus Only)	Pr 85.XX	Pr 115.XX	Pr 145.XX	Pr 175.XX
Digital I/O Parameters (Apps, Apps Plus & ST Plus Only)	Pr 86.XX	Pr 116.XX	Pr 146.XX	Pr 176.XX
Status Parameters	Pr 88.XX	Pr 118.XX	Pr 148.XX	Pr 178.XX
General Parameters	Pr 90.XX	Pr 120.XX	Pr 150.XX	Pr 180.XX
Fast Access Parameters	Pr 91.XX	Pr 121.XX	Pr 151.XX	Pr 181.XX

The fieldbus interface module reads and writes data directly to and from the internal registers in an SM-Register. The fieldbus interface module can read data from and write data to an SM-Register installed in any slot in the drive, simply by specifying the target parameter as shown in Table 8.13.

**NOTE** If a single SM-Register module is installed to the drive, normal SM-Register parameter references can be used without problem, as the fieldbus will automatically divert them to the SM-Register.

### 8.3.1 Example configuration 1

Consider a drive with the following configuration:

- Slot 1 - Vacant
- Slot 2 - Application Processor
- Slot 3 - SM-PROFIBUS-DP module

If a parameter read request comes over the PROFIBUS-DP network to read Pr 71.08, this will be re-directed to the Application Processor in the lowest slot number, i.e. slot 2. The value in \_Q08% from slot 2 will be returned.

If a parameter read request comes over the PROFIBUS-DP network to read Pr 131.08, this will be sent straight to the Application Processor in slot 2. The value in \_Q08% from slot 2 will be returned.

If a parameter read request comes over the PROFIBUS-DP network to read Pr 101.08, this will be sent straight to the Application Processor in slot 1. As there are no Application Processor modules installed in slot 1, an error message will be returned, indicating that the parameter does not exist.

If, for example, an SM-DeviceNet module is installed to slot 1, you could use the direct slot parameter reference to read or write data, giving a simple communications gateway between DeviceNet and Profibus-DP.



### 8.3.2 Example configuration 2

Consider a Unidrive SP with the following configuration:

- Slot 1 - Application Processor
- Slot 2 - Application Processor
- Slot 3 - SM-PROFIBUS-DP module

If a parameter read request comes over the PROFIBUS-DP network to read Pr **71.08**, this will be re-directed to the Application Processor in the lowest slot number, i.e. slot 1. The value in \_Q08% from slot 1 will be returned.

If a parameter read request comes over the PROFIBUS-DP network to read Pr **131.08**, this will be sent straight to the Application Processor in slot 2. The value in \_Q08% from slot 2 will be returned.

If a parameter read request comes over the PROFIBUS-DP network to read Pr **101.08**, this will be sent straight to the Application Processor in slot 1. The value in \_Q08% from slot 1 will be returned.

**NOTE** If more than one Application Processor is installed to the drive, it is best to access the module parameters using the direct slot parameter references. If normal Application Processor parameters are used, and the Application Processor is removed from slot 1, these parameter references will be re-directed to slot 2 instead.

### 8.3.3 Example configuration 3

Consider a drive with the following configuration:

- Slot 1 - Application Processor
- Slot 2 - Application Processor
- Slot 3 - Application Processor

The Application Processor does not allow reading from or writing to Direct slot parameter references, so data transfer must be done another way.

If the Application Processor in slot 1 wanted to read register Pr **71.08** from the Application Processor in slot 3, this must be done through one of the 32 bit application parameters (Pr **20.21**-Pr **20.40**). The Application Processor in slot 1 must write the register data to the application parameter. The Application Processor in slot 3 can then read that parameter.

Code in slot 1 :

#20.21 = #71.08

Code in slot 3 :

#71.08 = #20.21

This data can also be accessed by the Application Processor in slot 2.

**NOTE** The Application Processor cannot access the Pr **1xx.XX** internal menus. It must access them directly by the parameter reference number. e.g. if you wanted to read parameter Pr **104.35** you would need to use Pr **74.35**. Therefore, it is not possible to access these parameters/registers via inter-option communications.

The example code above will not synchronise the updating of parameter Pr **71.08** in the 2 modules. For more information refer to section 11 *Inter-option synchronisation* on page 131.

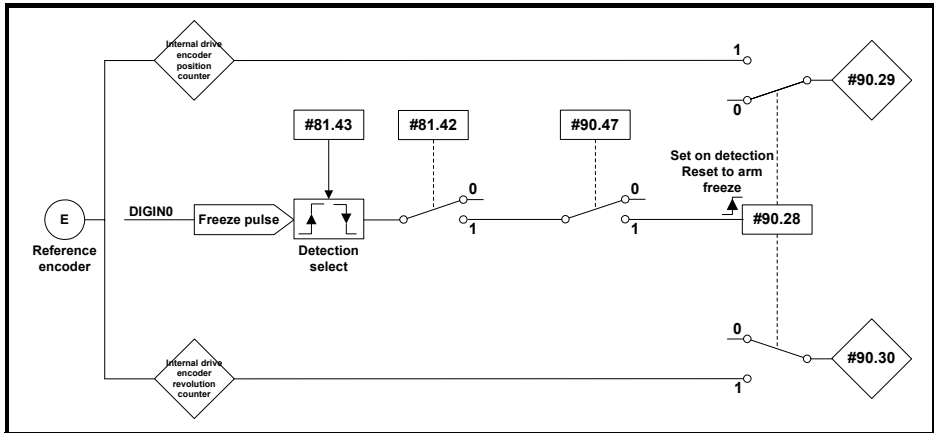
If the Application Processor are connected on the same CTNet network then this can be done through cyclic data. For more information on CTNet refer to the CTNet *User Guide*.

## 9 Freeze and Marker

### 9.1 Freeze input

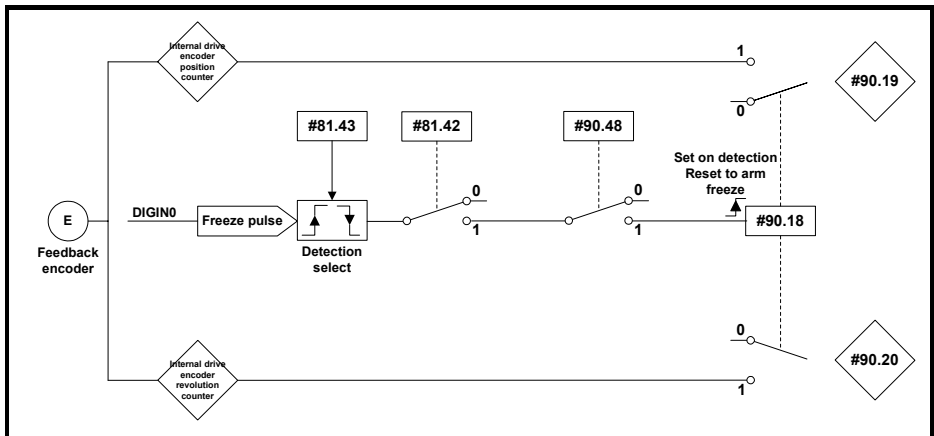
Digital Input 0 (DIGIN0) can be used to 'freeze' the reference and feedback encoder counters.

**Figure 9-1 SM-Register Reference Freeze Input**



The encoder revolution counter is cached into Pr **90.30** and the encoder position is cached into Pr **90.29**.

**Figure 9-2 SM-Register Feedback Freeze Input**



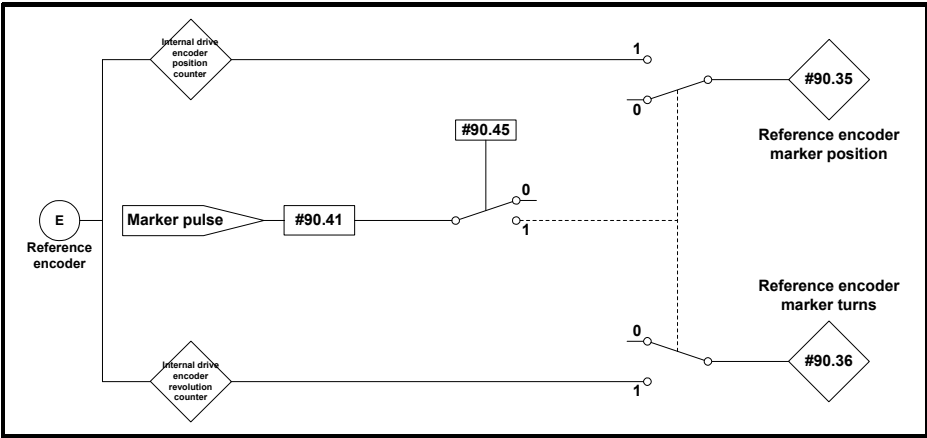
The encoder revolution counter is cached into parameter Pr **90.20** and the encoder position is cached into Pr **90.19**.

**NOTE** When freezing the SM-Universal Encoder Plus data, the Application Processor's freeze input cannot be used. The freeze should come from the SM-Universal Encoder Plus module. Refer to the *SM-Universal Encoder Plus User Guide* for more information on using its freeze with the SM-Register.

## 9.2 Marker pulse

The SM-Register is able to cache the position and revolution count at the point when a Marker pulse is seen on the reference or feedback encoders.

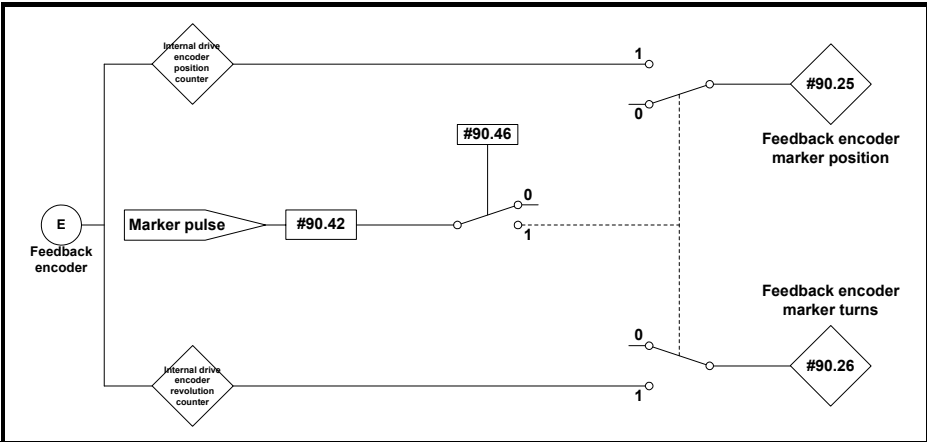
**Figure 9-3 SM-Register reference marker**



The marker position is cached into Pr **90.35** and the marker revolution counter is cached into Pr **90.36**.

The user sets Pr **90.41** to a zero and the drive sets Pr **90.41** to a 1 EVERY time a marker is detected. The marker data must be consumed before the next marker pulse.

**Figure 9-4 SM-Register feedback marker**



The marker position is cached into Pr **90.25** and the marker revolution counter is cached into Pr **90.26**.

The user sets Pr **90.42** to a zero and the drive sets Pr **90.42** to a 1 EVERY time a marker is detected. The marker data must be consumed before the next marker pulse.

## 10 CTSync

### 10.1 Overview

The SM-Register may be used to synchronise two or more drives. This will ensure that the drives run their internal functions at exactly the same frequency and time meaning all actions are performed at the same instant.

Also, 3 data values can be passed from one module (the Master) to others (Slaves) on the CTSync network. This comprises 2 x signed 32bit integers and 1 x Unsigned 8bit integer.

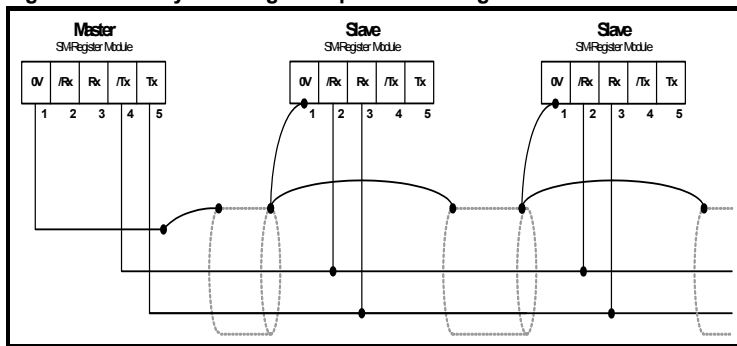
Only one SM-Register / Motion Processor should be configured as the Master and all others configured as Slaves if they need to participate in the CTSync scheme. The Master generates reference data which is transmitted to all Slaves on the network. The Master can be set to operate as a Slave, if for instance two drives need to be synchronised. In this case the Master will be generating the reference data as well as following that reference data. The slave will also be following that reference data.

### 10.2 Connections

CTSync operates via a connection between the EIA-485 ports of the SM-Register / Motion processors on the network in either 2-wire or 4-wire. Refer to section 3.7 *EIA-485 connections* on page 15 for information on how to connect the SM-Register Module EIA-485 ports.

To simplify wiring the Slave transmit and Master receive signal line connections can be omitted in 4-wire mode (See Figure 10-1). This is because the Master does not receive a response from the Slave.

**Figure 10-1 CTSync wiring example for SM-Register**



### 10.3 Limitations

- Only one CTSync Master is permitted on the Network.
- All CTSync Master and Slave Pos tasks must be set to the same update time (parameter Pr **81.12**).
- 8 nodes maximum for 2-wire and 16 nodes maximum for 4-wire before line repeaters are required.
- Maximum cable length of EIA-485 network is 1200 metres.

### 10.4 CTSync function blocks

Six Function Blocks are available when using the CTSync functionality of the SM-Register. These are listed below.

#### 10.4.1 CTSYNCSetMasterReferences

```
CTSYNCSetMasterReferences(Reference1%, Reference2%, AuxiliaryReference%)
```

Input Arguments	
Argument name	Range
Reference1	Signed 32-bit
Reference2	Signed 32-bit
AuxiliaryReference	Unsigned 8-bit

This function block allows the CTSync Master to write reference data to all CTSync Slaves on the network. This command will have no effect if used in a CTSync Slave.

#### 10.4.2 CTSYNCGetSlaveReferences

```
(Reference1%, Reference2%, AuxiliaryReference%, Status%) =  
CTSYNCGetSlaveReferences()
```

Output Arguments	
Argument name	Range
Reference1%	Signed 32-bit
Reference2%	Signed 32-bit
AuxiliaryReference%	Unsigned 8-bit

This function block allows the CTSync to read the reference data created by the CTSync Master. It can be used on both the Master and Slaves.

The block returns the values of the last Master references received.

**Output Arguments:**

- Status%:
- 1: OK
  - 0: Zero or too few data received.
  - 1: More bytes than expected received.
  - 2: Checksum error in received data.
  - 3: Not in CTSync mode.

If Status does not contain OK then Reference1%, Reference2% and AuxiliaryReference% are not modified.

#### 10.4.3 CTSYNCSetupOutputChannel

```
Result% = CTSYNCSetupOutputChannel(Channel%, Menu%, Parameter%)
```

This function block configures one of the 3 channels to route any data passed to it to a specified drive parameter.

**Input Arguments:**

Channel%: 1,2 or 3 for the 3 available "channels".  
 Menu%: Drive menu number to write to.  
 Parmeter%: Drive parameter number to write to.

**Output Arguments:**

Result%: 1: Operation succeeded.  
 -1: Invalid channel specified.  
 -3: Channel configuration being run by another task.  
 -4: Parameter does not exist or is read-only

**10.4.4 CTSYNCEnableOutputChannel**

```
Result% = CTSYNCEnableOutputChannel(Channel%)
```

This function block allows the specified channel to write it's data to the drive at the start of each motion engine sample period.

**Input Arguments:**

Channel%: 1,2 or 3 for the 3 available "channels".

**Output Arguments:**

Result%: 1: Operation succeeded.  
 0: Channel is not set up correctly.  
 -1: Invalid channel specified.  
 -3: Channel configuration being run by another task.

**10.4.5 CTSYNCDisableOutputChannel**

```
Result% = CTSYNCDisableOutputChannel(Channel%)
```

This function block causes the specified channel to cease writing it's data to the drive.

**Input Arguments:**

Channel%: 1,2 or 3 for the 3 available "channels".

**Output Arguments:**

Result%: 1: Operation succeeded.  
 -1: Invalid channel specified.  
 -3: Channel configuration being run by another task.

**10.4.6 CTSYNCWriteOutputChannel**

```
Result% = CTSYNCwriteOutputChannel(Channel%, value%)
```

This function block writes a value to the specified channel.

**Input Arguments:**

Channel%: 1,2 or 3 for the 3 available "channels".  
 Value%: Value to write.

**Output Arguments:**

Result%: 1: Operation succeeded.  
 0: Channel is not set up correctly.  
 -1: Invalid channel specified.  
 -3: Channel configuration being run by another task.

If the value to be written is over-range for the parameter, the drive will be tripped (tr44) if **Pr 81.14=1** and **Pr 81.17=1**, or the value will be clamped if either of these is set to zero.

**10.5 Motion engine**

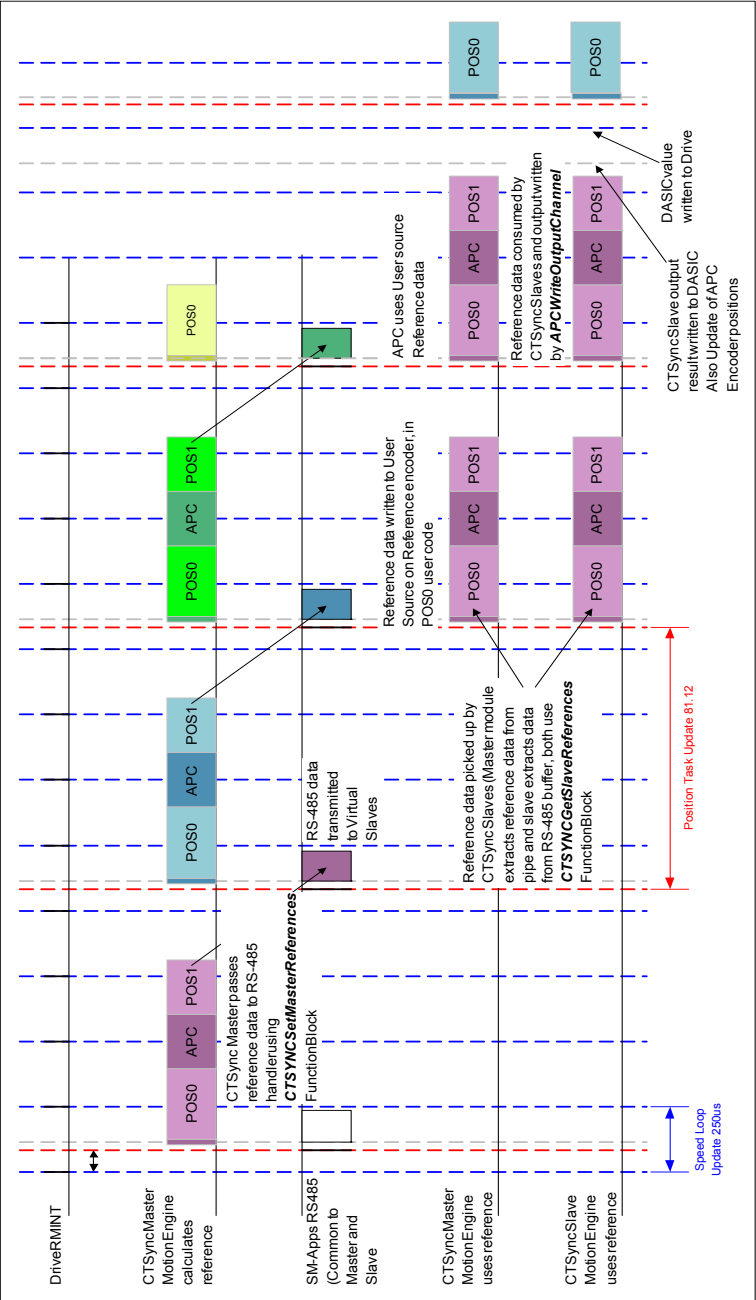
The motion engine tasks for each motion engine sample are shown below:

1. The Master motion engine calculates the reference.
2. The Master passes the reference data to the RS-485 handler by using the **CTSYNCSetMasterReferences** function block.
3. The RS-485 data is transmitted to the Slave(s).
4. The reference data is retrieved by the Slave(s) using the **CTSYNCGetSlaveReferences** function block.

5. The reference data is output by the **CTSYNCWriteOutputChannel** function block (if required) to the channels specified.
6. The Slave output values are written to the Drive parameters via the Drive's ASIC.
- For more information on the timings refer to Figure 10-2 *Motion Engine timing* .

Safety Information
Introduction
Installation
Getting started
SM-Register Functionality
Parameters
DPL Programming
Communications
Freeze and Marker
<b>CTsync</b>
Inter-option synchronisation
Diagnostics
Quick Reference
Index

Figure 10-2 Motion Engine timing





## 10.6 Virtual master example

### 10.6.1 Example master code

The following code shows how the master would be set up to generate reference data and send the data to the slaves. It also shows that the master acts as a slave by using the reference data it has generated. This is done with the

**CTSYNCGetSlaveReferences** function block.

The example generates Virtual Master data as a triangular ramp and also implements a Virtual Slave which outputs the data to the drive.

```
Initial
{
    #81.06=25 //CTSync Master
    #81.12=2 //Pos task at 500us
    REINIT

    //Set up Virtual Master Ramp Reference.
    Ramp% = 0

    //Configure Slave output channel 1 to menu 18, parameter 11.
    CTSYNCSetupOutputChannel(1,18,11)

    //Enable the configured output channel.
    CTSYNCEnableOutputChannel(1)
}

POS0
{
    //Perform Slave function first, so timing is identical between master and slave:
    (Slaveref1%,Slaveref2%,Auxref%, Status%) = CTSYNCGetSlaveReferences()

    //Status% should be checked here to ensure the data was received correctly.
    if Status% = USR_VALUE_OK then

        //Output to the drive at the start of next motion period.
        CTSYNCWriteOutputChannel(1,Slaveref1%)

    endif

    //Perform Master function, in this case a ramp:
    Ramp% = Ramp% + 1

    //Apply reset to ramp profile if required
    if Ramp% > 999 then

        //wrap back to zero
        Ramp% = 0
    endif

    //Pass Reference Data to Slaves. Note only reference1 used.
    CTSYNCSetMasterReferences(Ramp%,0,0)
}
```

Safety
Information
Introduction
Installation
Getting started
SM-Register Functionality
Parameters
Programming
DPL
Communications
Freeze and Marker
CTSync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

## 10.6.2 Example slave code

The following code shows how the slave(s) would read the reference data generated by the Virtual Master.

```
Initial
{
    #81.06=26 //CTSync Slave
    #81.12=2  //Pos task at 500us
    REINIT

    //Set up Virtual Master Ramp Reference.
    Ramp% = 0

    //Configure Slave output channel 1 to menu 18, parameter 11.
    CTSYNCSetupOutputChannel(1,18,11)

    //Enable the configured output channel.
    CTSYNCEnableOutputChannel(1)
}

POS0
{
    //Perform Slave function first, so timing is identical between master and slave:
    (Slaveref1%,Slaveref2%,Auxref%, Status%) = CTSYNCGetSlaveReferences()

    // Check the data was received in good order.
    if Status% = USR_VALUE_OK then

        //Output to the drive at the start of next motion period.
        CTSYNCwriteOutputChannel(1,Slaveref1%)

    endif
}
```

# 11 Inter-option synchronisation

## 11.1 Overview

The Inter-option Module Synchronisation scheme provides a mechanism to allow position control tasks on one or more modules to be synchronised to position control tasks on another module on the same drive without the need for external wiring. Synchronisation is achieved by one module producing a trigger signal and one or more modules consuming the trigger signal. These modules are referred to as Producers and Consumers. The trigger signal is used to alter the scheduling of the POS engine. If the module is a Producer the trigger may be provided at a rate of 250µs, 500µs, 1ms, 2ms, 4ms or 8ms i.e. Producer modules generate the trigger every POS period, as specified in Pr **81.12**.

This scheme is in addition to the CTSync scheme (refer to section 10 CTSync on page 124 for more information), although it can be used in conjunction with CTSync to synchronise modules in more than one drive (see section 11.3 *Inter-option synchronisation example 2* on page 134).

The module, when a Consumer, can run its Position Control tasks at the same rate or quicker than the Synchronisation Signal (trigger) from the Producer.

If a module is a Consumer and its position control task is set to run slower than that of the Synchronisation Signal (trigger) from the Producer, then it will be impossible for it to know the phase of the Producer, so it will not attempt to synchronise to the Producer's signal (see section 11.3 *Inter-option synchronisation example 2* on page 134).

## 11.2 Inter-Option synchronisation example 1

This example shows a scenario where the POS0 tasks of the modules in a drive need to be synchronised.

The following table shows the settings for each of the 3 modules:

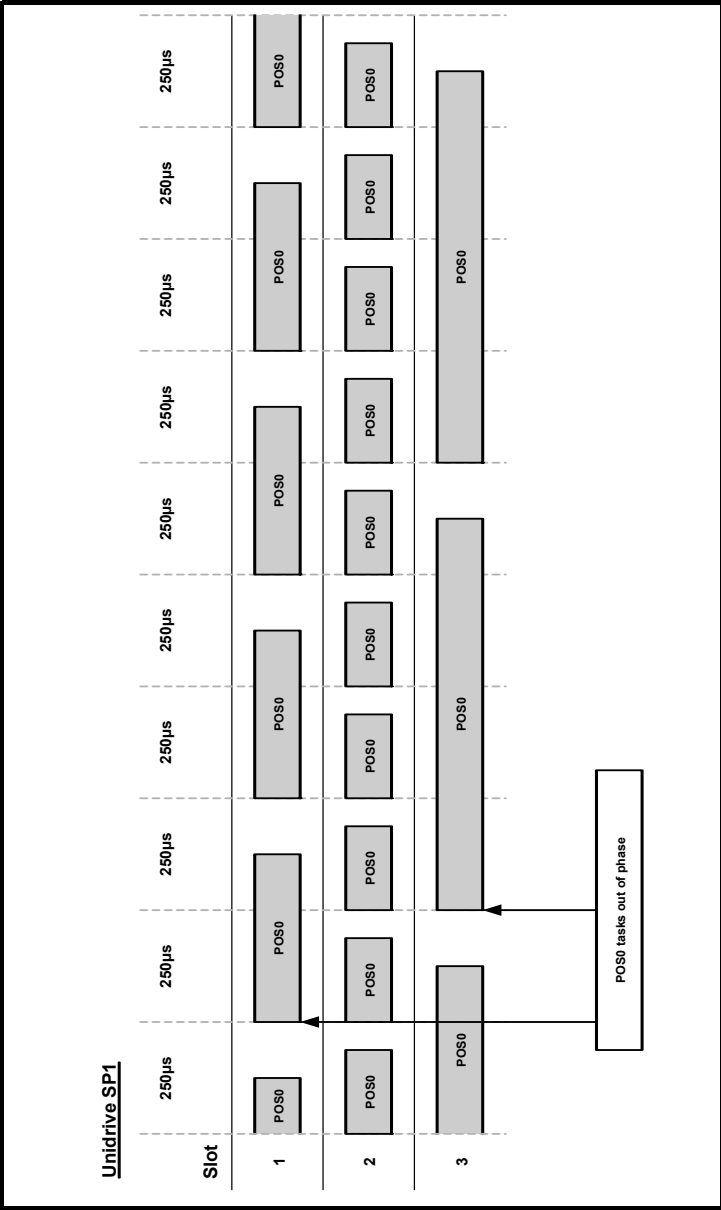
**Table 11.1 Parameter settings**

	Parameter	Value	Description
Slot 1	Pr <b>81.06</b>	1 (default)	Not in CTSync scheme.
	Pr <b>81.12</b>	2	500µs position control task period.
	Pr <b>91.21</b>	2	Inter-Option Sync Consumer.
	Pr <b>91.22</b>	6	Inter-Option Sync Consumer status achieved.
Slot 2	Pr <b>81.06</b>	1 (default)	Not in CTSync scheme.
	Pr <b>81.12</b>	1	250µs position control task period.
	Pr <b>91.21</b>	2	Inter-Option Sync Consumer.
	Pr <b>91.22</b>	6	Inter-Option Sync Consumer status achieved.
Slot 3	Pr <b>81.06</b>	1 (default)	Not in CTSync scheme.
	Pr <b>81.12</b>	3	1ms position control task period.
	Pr <b>91.21</b>	1	Inter-Option Sync Producer.
	Pr <b>91.22</b>	5	Inter-Option Sync Producer status achieved.

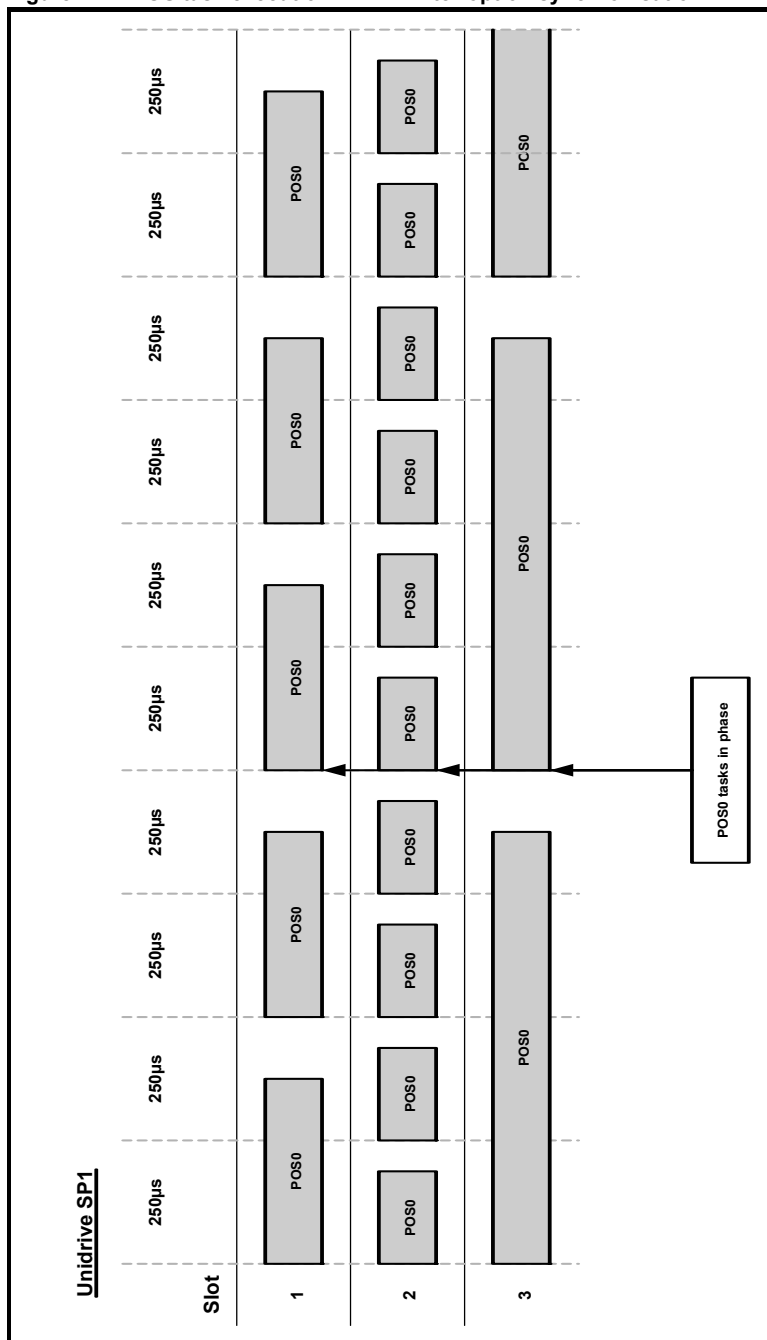
The italicised Pr **91.22** is a status parameter and does not require setting. It shows whether the module has achieved the Inter-Option functionality specified in Pr **91.21**. For more information refer to Pr **91.12**.

Before setting the parameters above, the POS tasks may be executing out of phase with each other as shown in Figure 11-1 *POS Task execution BEFORE inter-option synchronisation* . After setting the parameters to the values shown above the POS tasks will be executing in phase with each other as shown in Figure 11-2 *POS task execution AFTER inter-option synchronisation* .

**Figure 11-1 POS Task execution BEFORE inter-option synchronisation**



**Figure 11-2 POS task execution AFTER inter-option synchronisation**



# 11.3 Inter-option synchronisation example 2

This example shows a scenario where the POS0 tasks of the modules in 2 Unidrive SPs need to be synchronised.

For this to work a module on each drive needs to be configured to participate in the CTSync scheme (refer to section 10 CTSync on page 124 for more information). One module must be configured as the CTSync Master and one module configured as the CTSync Slave.

The following tables show the settings for each of the 6 modules:

**Table 11.2 Parameter settings for Unidrive SP 1**

	Parameter	Value	Description
Slot 1	Pr 81.06	1 (default)	Not in CTSync scheme.
	Pr 81.12	2	500µs position control task period.
	Pr 91.21	2	Inter-Option Sync Consumer.
	Pr 91.22	6	Inter-Option Sync Consumer status achieved.
Slot 2	Pr 81.06	1 (default)	Not in CTSync.
	Pr 81.12	1	250µs position control task period.
	Pr 91.21	2	Inter-Option Sync Consumer.
	Pr 91.22	6	Inter-Option Sync Consumer status achieved.
Slot 3	Pr 81.06	25 (default)	CTSync Master.
	Pr 81.12	3	1ms position control task period.
	Pr 91.21	1	Inter-Option Sync Producer.
	Pr 91.22	5	Inter-Option Sync Producer status achieved.

**Table 11.3 Parameter settings for Unidrive SP 2**

	Parameter	Value	Description
Slot 1	Pr 81.06	1 (default)	Not in CTSync scheme.
	Pr 81.12	2	500µs position control task period.
	Pr 91.21	2	Inter-Option Sync Consumer.
	Pr 91.22	6	Inter-Option Sync Consumer status achieved.
Slot 2	Pr 81.06	26	CTSync Slave.
	Pr 81.12	3	1ms position control task period.
	Pr 91.21	1	Inter-Option Sync Producer.
	Pr 91.22	13	Inter-Option Sync Producer status achieved with DINT signal.
Slot 3	Pr 81.06	1 (default)	Not in CTSync scheme.
	Pr 81.12	4	2ms position control task period.
	Pr 91.21	2	Inter-Option Sync Consumer.
	Pr 91.22	2	Inter-Option Sync Consumer status not achieved (POS task too slow).

The italicised Pr **91.22** is a status parameter and does not require setting. It shows whether the module has achieved the Inter-Option functionality specified in Pr **91.21**. For more information refer to Pr **91.12**.

**NOTE** When a module is a CTSync Slave, it has to be the Inter-Option Synchronisation Producer on the drive it is installed to therefore Pr **91.21** will automatically be set to a value of 1.

Before setting the parameters above, the POS tasks may be executing out of phase with each other as shown in Figure 11-3 *POS task execution BEFORE inter-option synchronisation with CTSync* . After setting the parameters to the values shown above the POS tasks should be executing in phase with each other as shown in Figure 11-4 *POS task execution AFTER inter-option synchronisation with CTSync* .

Safety Information
Introduction
Installation
Getting started
SM-Register Functionality
Parameters
DPL Programming
Communications
Freeze and Marker
CTSync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

Figure 11-3 POS task execution BEFORE inter-option synchronisation with CTSync

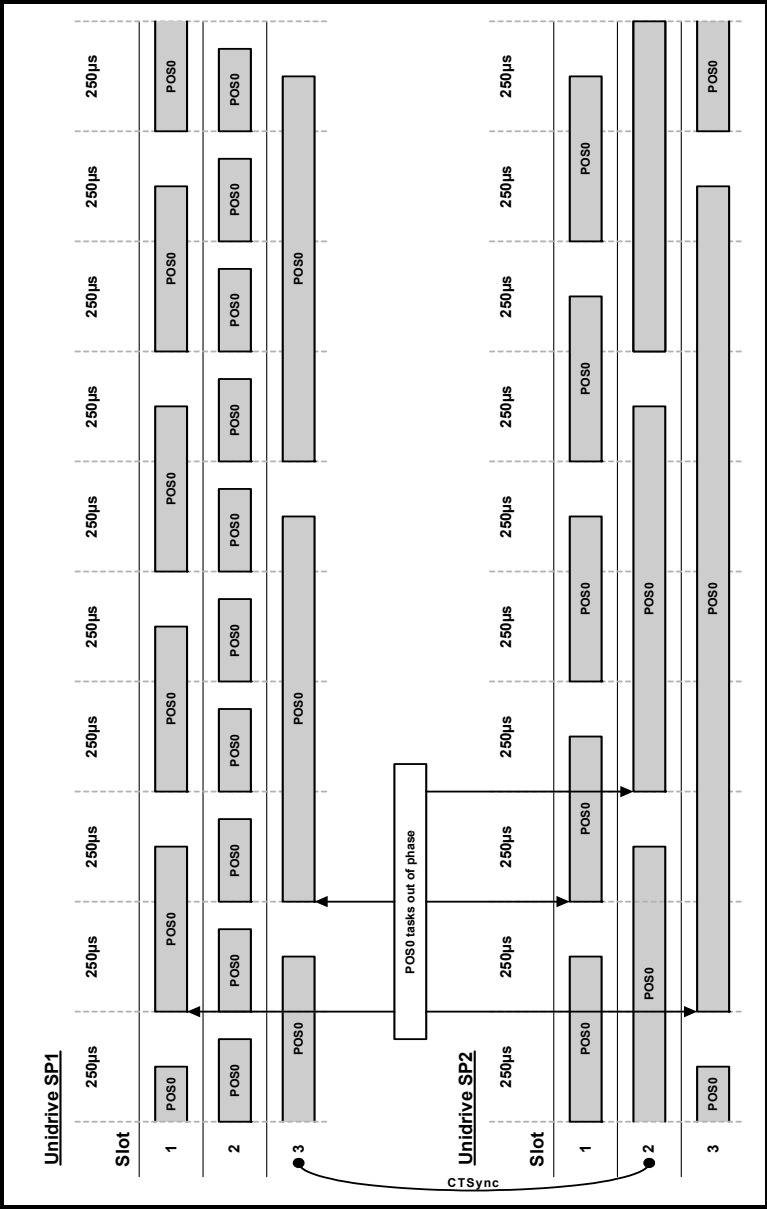
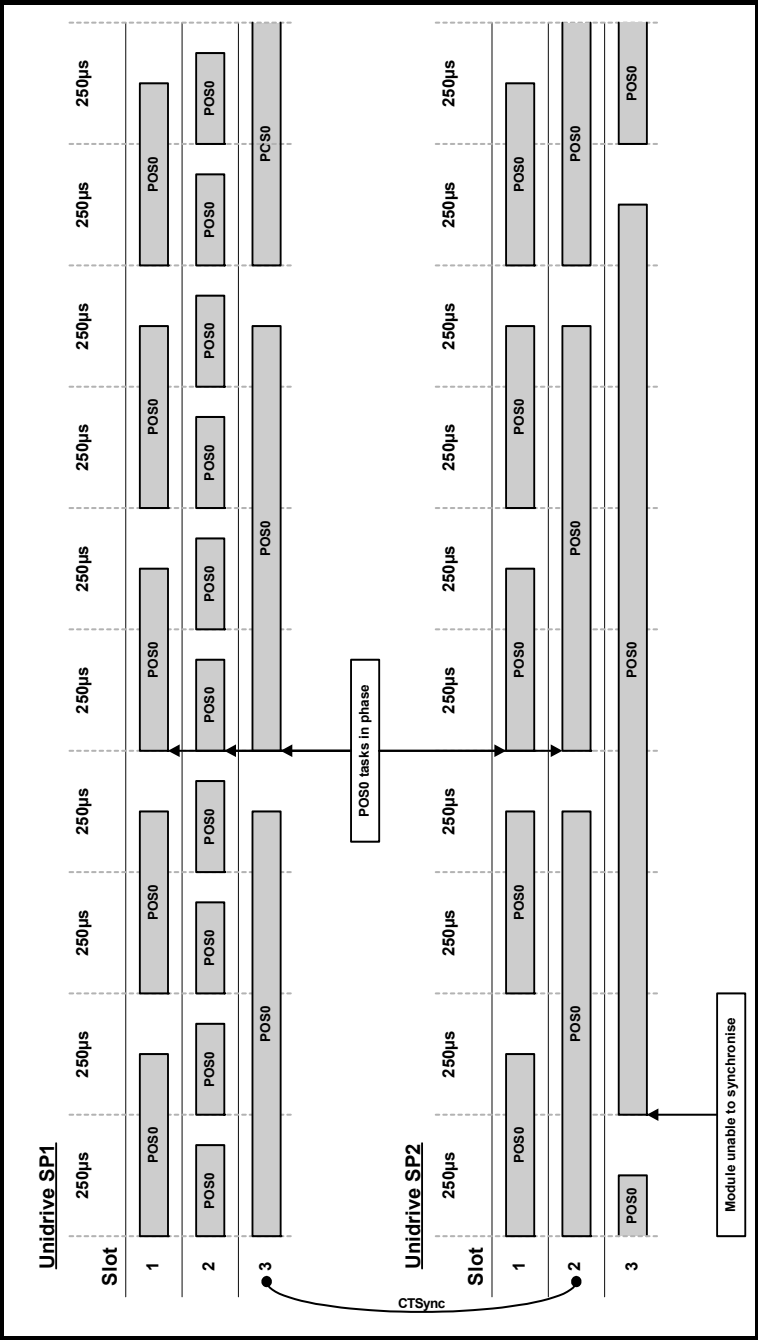




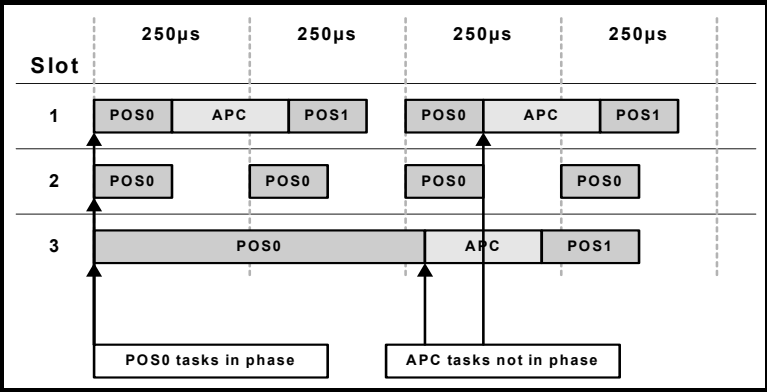
Figure 11-4 POS task execution AFTER inter-option synchronisation with CTSync



# 11.4 Position control tasks

The position control tasks consist of POS0, APC and POS1. The tasks are run in this order (POS0 runs first followed by the APC which is followed by POS1). Because the POS0 and APC tasks may not take the same time to execute on different modules the APC and POS1 tasks cannot be phased. Therefore when using Inter-option synchronisation the user must be aware that the position tasks as a whole for each module will start in phase, not the individual tasks. Figure 11-5 *Position control tasks* shows an example of this.

**Figure 11-5 Position control tasks**



# 12 Diagnostics

This chapter details the following:

- Run-time errors and trip codes
- Handling of run-time errors
- CTNet network status
- Support

## 12.1 Run-time errors

A run-time error is an error which occurs in a specific operation of the SM-Register. It can happen as a result of an error in the execution of the user's DPL program (such as trying to write to a parameter that doesn't exist, or trying to divide a value by zero), a misconfiguration (such as incorrect CTNet setup) or a system error such as processor overload or *watchdog* time-out.

The action taken when an error occurs may be one or more of the following:

- User program may be halted or prevented from starting
- Drive may be tripped on **SLx.Er** (where x is the slot number) with the run-time error code placed into Pr **xx.50**
- Drive may be tripped on another **SLx.\*\*\*** code.
- The DPL **ERROR** task may be executed (if it exists).

Which of these occurs depends upon the type of error and the setting of the global run-time trip enable parameter Pr **xx.14**. This is detailed in section 12.3 *SM-Register run-time error codes* , below.

## 12.2 Drive display trip codes

The table below shows the possible trip codes that will be displayed on the drive when an error is detected in the SM-Register which instigates a drive trip. Remember, not all run-time errors instigate a drive trip.

Table 12.1 Drive trip display codes

Slot Where Trip Was Initiated			Fault	Description
Slot 1	Slot 2	Slot 3		
SL1.HF	SL2.HF	SL3.HF	Hardware fault	The drive has detected that an option module is present, but is unable to communicate with it.
SL1.tO	SL2.tO	SL3.tO	Watchdog timeout	Indicates a user program which has utilised the <i>watchdog</i> feature has failed to issue the WDOG command within 200ms.
SL1.Er	SL2.Er	SL3.Er	Error	Run-time trip generated by the SM-Register either due to a user DPL program error or some other event. The actual error code is placed into Pr <b>xx.50</b> .
SL1.nF	SL2.nF	SL3.nF	Not installed	Module was disconnected while operational, or module has crashed. This trip will also occur if a drive slot is configured for a SM-Register, but the module is not installed in the slot.
SL1.dF	SL2.dF	SL3.dF	Different installed	This trip will occur when an SM-Register is installed to a slot previously occupied by another solutions module, or is installed to a previously unused slot.

## 12.3 SM-Register run-time error codes

If the SM-Register detects an error during operation the error code will be placed in the following parameter:

Pr 81.50	SM-Register error code		
Access	RO	Range	0 to 255
Default	N/A	Update Rate	On error

For certain errors the user may select if the drive should trip as well. This is configured with the Global Run-time Trip enable parameter:

Pr 81.14	Global run-time trip enable		
Access	RW	Range	0/1
Default	0	Update Rate	N/A

If set to 1 (On), the drive will trip on ALL run-time errors.

The table below shows the error codes and their meaning as well as if the drive will trip, the User program will stop and whether the DPL ERROR task will run.

### Notes:

- **“May”** under Drive Trip indicates that the drive will only trip if the global run-time trip enable parameter is set
- **“Not Run”** under Program Halted indicates that the error occurs at initialisation and the program will not be started.

**Table 12.2 SM-Register error codes**

Error Code	Reason	Trip Drive?	ERROR task?	Prog Halted?
39	User program stack overflow.	Yes	No	Yes
40	Unknown error - please contact supplier.	Yes	No	Yes
41	Parameter does not exist. User has attempted to read or write a non-existent parameter in the DPL program.	May	Yes	Yes
42	Attempt to write to a read-only parameter.	May	Yes	Yes
43	Attempt to read from a write-only parameter.	May	Yes	Yes
44	Parameter value out of range. (User has written an illegal value to a parameter within a DPL program.) If parameter #MM.17=0 the value written will be automatically limited and no error will occur.	May	Yes	Yes
45	Invalid synchronisation modes.	Yes	No	Not Run
46	Unused.	N/A	N/A	N/A
48	RS485 not in user mode. Occurs if user attempts to use a user-mode RS485 DPL command but the RS485 port is not in a user-mode.	Yes	Yes	Yes
49	Invalid RS485 configuration. For example, invalid mode.	Yes	Yes	Yes
50	Maths error - divide by zero or overflow.	May	Yes	Yes
51	Array index out of range. E.g. arr%[20] where arr% has only been DIMensioned to 19 elements.	May	Yes	Yes
52	Control word user trip. Instigated by setting the trip bit in the control word <b>#90.11</b> .	Yes	No	No
53	DPL program incompatible with target. For example, downloading a program compiled for UD70.	Yes	N/A	N/A

**Table 12.2 SM-Register error codes (Continued)**

Error Code	Reason	Trip Drive?	ERROR task?	Prog Halted?
54	DPL task overrun. This occurs if the DPL code within a real-time task (e.g. POS0) cannot be completed in time. Use parameter #88.02 to identify the task in which this error occurred. Check that the task scheduling rate is correct and that there are no loops in the task. This can also occur as a result of external influences such as a large burst of data coming in over CTNet. This problem may be overcome by changing the CTNet priority so that it is lower than the POS tasks. This, however, may cause the CTNet task to be starved. Refer to parameter <i>Pr 81.44</i> on page 78 for further information.	May	Yes	Yes
55	Invalid encoder configuration. Only applies to system file V01.02.01 or earlier.	Yes	N/A	N/A
56	Invalid timer unit configuration.	Yes	Yes	Yes
57	Function block does not exist.	Yes	Yes	Not Run
58	Flash PLC Storage corrupt. Occurs at startup and will mean that the PLC register set (P/Q/T/U) and menu 20 will not have been restored. If this problem persists it may indicate a hardware failure so contact your supplier.	Yes	Yes	Not Run
59	Drive rejected application module as Sync master.	Yes	Yes	Yes
60	CTNet hardware failure. Please contact your supplier.	May	No	No
61	CTNet invalid configuration. Check all configuration parameters	May	No	No
62	CTNet invalid baud-rate. Check #MM.24 and network connections.	May	No	No
63	CTNet invalid node ID. Check #MM.23.	May	No	No
64	Digital Output overload. Both digital outputs will be taken inactive when this occurs and will remain inactive until the error condition is cleared. The trip threshold is 20mA.	Yes	Yes	Yes
65	Invalid function block parameter(s). You have called a FB within a DPL program but one or more of the inputs are invalid.	Yes	Yes	Yes
66	User heap too large. The program has been compiled for a target that has more RAM than this one has. Occurs at startup.	Yes	No	Not Run
67	RAM file does not exist or a non-RAM file id has been specified.	Yes	Yes	Yes
68	The RAM file specified is not associated to an array.	Yes	Yes	Yes
69	Failed to update drive parameter database cache in Flash memory.	Yes	No	Not Run
70	User program downloaded while drive enabled. Will occur if #MM.37 = 1 and a program is downloaded.	May	No	Yes
71	Failed to change drive mode.	Yes	No	Yes
72	Invalid CTNet buffer operation.	Yes	Yes	Yes
73	Fast parameter initialisation failure.	Yes	No	No
74	Over-temperature.	Yes	Yes	Yes
75	Hardware unavailable. The user program attempted to access unavailable hardware. e.g. if access is made to digital I/O, RS485 port or CTNet on SM-Applications Lite module.	Yes	Yes	Yes

Safety Information
Introduction
Installation
Getting started
SM-Register
Parameters
DPL Programming
Communications
Freeze and Marker
CTSync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

**Table 12.2 SM-Register error codes (Continued)**

Error Code	Reason	Trip Drive?	ERROR task?	Prog Halted?
76	Module type cannot be resolved. Module is not recognised.	Yes	No	Not Run
77	Inter-Option module comms error with module in slot 1.	Yes	Yes	Yes
78	Inter-Option module comms error with module in slot 2.	Yes	Yes	Yes
79	Inter-Option module comms error with module in slot 3.	Yes	Yes	Yes
80	Inter-Option module comms error with module unknown slot.	Yes	Yes	Yes
81	<ul style="list-style-type: none"> <li>Slot selected as the reference or feedback does not contain a position option module.</li> <li>Attempt to change the Reference source or the Feedback source in more than one task.</li> </ul> APC internal error. See Pr <b>81.38</b> . This may be caused by one of the following: <ul style="list-style-type: none"> <li>CAM table too small.</li> <li>A change of too many CAM segments has occurred at the CAM table input.</li> <li>CAM is selected but size is zero.</li> <li>CAM absolute mode selected and Reset Index or Reset Position in segment is out of range.</li> </ul>	May	Yes	Yes
82	Communications to drive faulty.	May	Yes	Yes

## 12.4 Handling run-time errors with the ERROR task

Certain run-time errors will cause the DPL ERROR task to be invoked if it exists. This provides a convenient way to safely handle the error condition and take whatever action is necessary, such as a controlled stop of the system or signalling of an alarm.

When an ERROR task runs, all other DPL tasks will have been stopped. Therefore the ERROR task has exclusive execution rights. Once the ERROR task has completed, the DPL program ends and no other DPL tasks operate (though it is possible to reset and restart the program - more details on this below).

**NOTE** Drive trips do not cause the ERROR task to run. Only certain DPL program errors do.

Within the ERROR task all standard DPL commands may be used as well as most function blocks. All drive and SM-Register parameters can be accessed.

The run-error code can be determined using this parameter:

Pr 88.01	Error status / reset		
Access	RW	Range	0 to 9999
Default	N/A	Update Rate	On error

This parameter has two purposes - when read it will return the run-time error code the same as Pr **81.50** (note - it will not return drive trip codes). The parameter is cleared to zero on reset and when the user program execution is started.

When the parameter is written to with a value of 1070 the SM-Register will initiate a warm-restart of the drive and any other options. This can be used to restart the user program (providing auto-run Pr **81.13**=1) and clear any drive trip. This reset action can be performed at any time, not just after a run-time error or in an ERROR task



Writing 1070 to Pr **88.01** will result in any drive trip being automatically cleared as well as resetting all installed options in the Unidrive SP.

The task that caused a run-time error can be determined by reading Pr **88.02**, as previously described.

If the user wishes to trip the drive (if it hasn't already been tripped) then write the appropriate trip code into Pr **10.38**.

## 12.5 Resource monitoring

Pr 88.03	POS resource monitoring		
Access	RW	Range	0/1
Default	0	Update Rate	Immediate

This parameter allows the user to enable or disable monitoring of the motion engine tasks free resource. When set to 1, parameters Pr **88.04** and Pr **88.05** become active. If set to zero, parameters Pr **88.04** and Pr **88.05** will read zero.

Pr 88.04	Free resource for motion engine tasks		
Access	RW	Range	0 to 95
Default	0	Update Rate	See Pr 81.12

This parameter indicates the percentage resource available for running the motion engine tasks. These tasks are CTSync, CTSync Output Channels, POS0, PLCOpen, APC, APC Output Channel and POS1. If this parameter value reaches zero a task overrun will occur. It is calculated every motion engine period and is displayed for the previous motion engine period

Pr 88.05	Motion engine peak resource detect		
Access	RW	Range	0 to 95
Default	0	Update Rate	See Pr 88.04

This parameter shows the peak that parameter Pr **88.04** reaches. It will show the lowest value that parameter Pr **88.04** reaches since the monitoring was enabled (parameter Pr **88.03**). It will give a realistic indication of the worst case available resources for the motion engine tasks so that the user can see how close the module has been to a motion engine task overrun.

Pr 88.06	CLOCK task resource monitoring		
Access	RO	Range	0/1
Default	NA	Update Rate	Immediate

This parameter allows the user to enable or disable monitoring of the CLOCK task free resource. When set to 1, parameters Pr **88.07** and Pr **88.08** become active. If set to zero, parameters Pr **88.07** and Pr **88.08** will read zero

<b>Pr 88.07</b>	<b>Free resource for clock task</b>		
<b>Access</b>	RO	<b>Range</b>	0 to 95
<b>Default</b>	NA	<b>Update Rate</b>	See Pr 81.11

This parameter indicates the percentage resource available for running the Clock task. If this parameter value reaches zero a task overrun will occur. It is calculated every Clock period and is displayed for the previous motion engine period.

<b>Pr 88.08</b>	<b>Clock task peak resource detect</b>		
<b>Access</b>	RO	<b>Range</b>	0 to 95
<b>Default</b>	NA	<b>Update Rate</b>	See Pr 81.11

This parameter shows the peak that parameter Pr 88.07 reaches. It will show the lowest value that parameter Pr 88.07 reaches since the monitoring was enabled (parameter Pr 88.06). It will give a realistic indication of the worst case available resources for the Clock task so that the user can see how close the module has been to a Clock task overrun.

## 12.6 Support

The information from the parameters described below should always be noted before contacting your supplier for technical support.

### 12.6.1 Module firmware

<b>Pr 81.02</b>	<b>Firmware - major version</b>		
<b>Access</b>	RO	<b>Range</b>	00.00 to 99.99
<b>Default</b>	N/A	<b>Update Rate</b>	N/A

<b>Pr 81.51</b>	<b>Firmware - minor version</b>		
<b>Access</b>	RO	<b>Range</b>	0 to 99
<b>Default</b>	N/A	<b>Update Rate</b>	N/A

The full version of the SM-Register firmware version can be read for the corresponding slot. The table below shows how to construct the full firmware version from these values.

**Table 12.3 Firmware version example**

Major version	Minor version	Firmware version
1.02	00	V01.02.00



# 13 Quick Reference

Refer to section 6 *Parameters* on page 67 for full details of these parameters.

**Table 13.1 Setup parameters**

Parameter	Description	Cross Ref	Range	Default
Pr 81.01	Module option code.	Section 6.3.1 on page 69	0-499	N/A
Pr 81.02	Module firmware version.	Section 6.3.1 on page 69	0-99.99	N/A
Pr 81.03	DPL program status.	Section 6.3.1 on page 69	0-3	N/A
Pr 81.04	Available system resource %.	Section 6.3.1 on page 69	0-100%	N/A
Pr 81.05	EIA-485 address.	Section 6.3.1 on page 69	0-255	11
Pr 81.06	EIA-485 Mode.	Section 6.3.1 on page 70	0-255	1
Pr 81.07	EIA-485 Baud rate.	Section 6.3.1 on page 71	0-9	4
Pr 81.08	EIA-485 Turn-around delay.	Section 6.3.1 on page 71	0-255ms	2ms
Pr 81.09	EIA-485 Tx Enable Delay.	Section 6.3.1 on page 71	0-1ms	0ms
Pr 81.10	DPL print routing.	Section 6.3.1 on page 72	0-1	0
Pr 81.11	Clock tick time (ms).	Section 6.3.1 on page 72	0-200ms	0
Pr 81.12	POS task schedule rate.	Section 6.3.1 on page 72	0-6	0
Pr 81.13	Enable autorun.	Section 6.3.1 on page 72	0-1	1
Pr 81.14	Global run-time trip enable.	Section 6.3.1 on page 72	0-1	0
Pr 81.15	Disable Reset on Trip Cleared.	Section 6.3.1 on page 73	0-1	0
Pr 81.16	Encoder Data Update Rate.	Section 6.3.1 on page 73	0-3	0
Pr 81.17	Enable parameter over-range trips.	Section 6.3.1 on page 73	0-1	0
Pr 81.18	Watchdog enable.	Section 6.3.1 on page 73	0-1	0
Pr 81.19	Save request.	Section 6.3.1 on page 73	0-1	0
Pr 81.20	Save on 'UU' trip.	Section 6.3.1 on page 74	0-1	0
Pr 81.21	Include menu 20 for save/restore.	Section 6.3.1 on page 74	0-1	0
Pr 81.22	CTNet token ring ID.	Section 6.3.1 on page 74	0-255	0
Pr 81.23	CTNet node address.	Section 6.3.1 on page 74	0-255	0
Pr 81.24	CTNet baud-rate.	Section 6.3.1 on page 74	0-3	1
Pr 81.25	CTNet sync setup (SSFF).	Section 6.3.1 on page 75	0-9999	0
Pr 81.26	CTNet Easy Mode 1 destination node (NNSS).	Section 6.3.1 on page 75	0-25503	0
Pr 81.27	CTNet Easy Mode 1 source parameter (MMPP).	Section 6.3.1 on page 75	0-9999	0
Pr 81.28	CTNet Easy Mode 2 destination node (NNSS).	Section 6.3.1 on page 75	0-25503	0
Pr 81.29	CTNet Easy Mode 2 source parameter (MMPP).	Section 6.3.1 on page 75	0-9999	0
Pr 81.30	CTNet Easy Mode 3 destination node (NNSS).	Section 6.3.1 on page 75	0-25503	0
Pr 81.31	CTNet Easy Mode 3 source parameter (MMPP).	Section 6.3.1 on page 75	0-9999	0
Pr 81.32	CTNet Easy Mode Slot 1 destination (MMPP).	Section 6.3.1 on page 75	0-9999	0

**Table 13.1 Setup parameters (Continued)**

Parameter	Description	Cross Ref	Range	Default
Pr 81.33	CTNet Easy Mode Slot 2 destination (MMPP).	Section 6.3.1 on page 75	0-9999	0
Pr 81.34	CTNet Easy Mode Slot 3 destination (MMPP).	Section 6.3.1 on page 75	0-9999	0
Pr 81.35	CTNet Sync EVENT task ID.	Section 6.3.1 on page 75	0-4	0
Pr 81.36	CTNet Diagnostic parameter.	Section 6.3.1 on page 76	-3-32767	N/A
Pr 81.37	Reject download if drive enabled.	Section 6.3.1 on page 76	0-1	0
Pr 81.38	APC Run-time trip.	Section 6.3.1 on page 76	0-1	0
Pr 81.39	Inter-option module Drive Synchronisation Status.	Section 6.3.1 on page 77	0-3	0
Pr 81.41	Indexer Go.	Section 6.3.1 on page 77	0-3	0
Pr 81.42	Freeze feedback encoder position.	Section 6.3.1 on page 77	0-1	0
Pr 81.43	Freeze invert.	Section 6.3.1 on page 77	0-1	0
Pr 81.44	Task priority level.	Section 6.3.1 on page 78	0-255	0
Pr 81.45	User Setup Parameter 1.	Section 6.3.1 on page 78	N/A	N/A
Pr 81.46	User Setup Parameter 2.	Section 6.3.1 on page 78	N/A	N/A
Pr 81.47	User Setup Parameter 3.	Section 6.3.1 on page 78	N/A	N/A
Pr 81.48	DPL line number of error.	Section 6.3.1 on page 78	32bit	0
Pr 81.49	User program ID.	Section 6.3.1 on page 79	16bit	0
Pr 81.50	Run-time error code.	Section 6.3.1 on page 79	0-255	0
Pr 81.51	Minor software revision.	Section 6.3.1 on page 79	0-99	N/A

**Table 13.2 Timer unit parameters**

Parameter	Description		Range	Default
Pr 85.01	Timer Unit Control Word.	Section 6.5 on page 80	13bit	N/A
Pr 85.02	Timer Unit Status Word.	Section 6.5 on page 81	0-3	N/A
Pr 85.03	Timer Unit 16-bit Timer Count.	Section 6.5 on page 81	16bit	N/A
Pr 85.04	Timer Unit Wrap-around Limit.	Section 6.5 on page 81	16bit	N/A
Pr 85.05	Timer Unit Timer Capture Cache.	Section 6.5 on page 82	16bit	N/A

**Table 13.3 Digital I/O parameters**

Parameter	Description		Range	Default
Pr 86.01	Digital Input 0.	Section 6.6 on page 82	0-1	N/A
Pr 86.02	Digital Input 1.	Section 6.6 on page 82	0-1	N/A
Pr 86.03	Digital Output 0.	Section 6.6 on page 82	0-1	N/A
Pr 86.04	Digital Output 1.	Section 6.6 on page 82	0-1	N/A
Pr 86.05	Digital Outputs 0 and 1.	Section 6.6 on page 82	0-3	N/A

**Table 13.4 Status parameters**

Parameter	Description		Range	Default
Pr <b>88.01</b>	Error Code / Reset.	Section 6.7 on page 83	0-9999	N/A
Pr <b>88.02</b>	Task In Error.	Section 6.7 on page 83	0-50	N/A
Pr <b>88.03</b>	POS Resource Monitoring.	Section 6.7 on page 83	0-1	0
Pr <b>88.04</b>	Free Resource for Motion Engine Tasks.	Section 6.7 on page 84	0-95	N/A
Pr <b>88.05</b>	Motion Engine Peak Resource Detect.	Section 6.7 on page 84	0-95	N/A
Pr <b>88.06</b>	CLOCK Task Resource Monitoring.	Section 6.7 on page 84	0-1	0
Pr <b>88.07</b>	Free Resource for Clock Task.	Section 6.7 on page 84	0-95	N/A
Pr <b>88.08</b>	Clock task peak resource detect.	Section 6.7 on page 84	0-95	N/A

**Table 13.5 Application processor error codes**

Error Code	Reason	Trip Drive?	ERROR task?	Prog Halted?
39	User program stack overflow.	Yes	No	Yes
40	Unknown error - please contact supplier.	Yes	No	Yes
41	Parameter does not exist. User has attempted to read or write a non-existent parameter in the DPL program.	May	Yes	Yes
42	Attempt to write to a read-only parameter.	May	Yes	Yes
43	Attempt to read from a write-only parameter.	May	Yes	Yes
44	Parameter value out of range. (User has written an illegal value to a parameter within a DPL program.) If parameter #MM.17=0 the value written will be automatically limited and no error will occur.	May	Yes	Yes
45	Invalid synchronisation modes.	Yes	No	Not Run
46	Unused.	N/A	N/A	N/A
48	RS485 not in user mode. Occurs if user attempts to use a user-mode RS485 DPL command but the RS485 port is not in a user-mode.	Yes	Yes	Yes
49	Invalid RS485 configuration. For example, invalid mode.	Yes	Yes	Yes
50	Maths error - divide by zero or overflow.	May	Yes	Yes
51	Array index out of range. E.g. arr%[20] where arr% has only been DIMensioned to 19 elements.	May	Yes	Yes
52	Control word user trip. Instigated by setting the trip bit in the control word #90.11.	Yes	No	No
53	DPL program incompatible with target. For example, downloading a program compiled for UD70.	Yes	N/A	N/A
54	DPL task overrun. This occurs if the DPL code within a real-time task (e.g. POS0) cannot be completed in time. Use parameter <b>#88.02</b> to identify the task in which this error occurred. Check that the task scheduling rate is correct and that there are no loops in the task. This can also occur as a result of external influences such as a large burst of data coming in over CTNet. This problem may be overcome by changing the CTNet priority so that it is lower than the POS tasks. This, however, may cause the CTNet task to be starved. Refer to parameter <i>Pr 81.44</i> on page 78 for further information.	May	Yes	Yes

**Table 13.5 Application processor error codes (Continued)**

Error Code	Reason	Trip Drive?	ERROR task?	Prog Halted?
55	Invalid encoder configuration. Only applies to system file V01.02.01 or earlier.	Yes	N/A	N/A
56	Invalid timer unit configuration.	Yes	Yes	Yes
57	Function block does not exist.	Yes	Yes	Not Run
58	Flash PLC Storage corrupt. Occurs at startup and will mean that the PLC register set (P/Q/T/U) and menu 20 will not have been restored. If this problem persists it may indicate a hardware failure so contact your supplier.	Yes	Yes	Not Run
59	Drive rejected Second Processor as Sync master.	Yes	Yes	Yes
60	CTNet hardware failure. Please contact your supplier.	May	No	No
61	CTNet invalid configuration. Check all configuration parameters.	May	No	No
62	CTNet invalid baud-rate. Check <b>#MM.24</b> and network connections.	May	No	No
63	CTNet invalid node ID. Check <b>#MM.23</b> .	May	No	No
64	Digital Output overload. Both digital outputs will be taken inactive when this occurs and will remain inactive until the error condition is cleared. The trip threshold is 20mA.	Yes	Yes	Yes
65	Invalid function block parameter(s). You have called a FB within a DPL program but one or more of the inputs are invalid.	Yes	Yes	Yes
66	User heap too large. The program has been compiled for a target that has more RAM than this one has. Occurs at startup.	Yes	No	Not Run
67	RAM file does not exist or a non-RAM file id has been specified.	Yes	Yes	Yes
68	The RAM file specified is not associated to an array.	Yes	Yes	Yes
69	Failed to update drive parameter database cache in Flash memory.	Yes	No	Not Run
70	User program downloaded while drive enabled. Will occur if <b>#MM.37</b> = 1 and a program is downloaded.	May	No	Yes
71	Failed to change drive mode.	Yes	No	Yes
72	Invalid CTNet buffer operation.	Yes	Yes	Yes
73	Fast parameter initialisation failure.	Yes	No	No
74	Over-temperature.	Yes	Yes	Yes
75	Hardware unavailable. The user program attempted to access unavailable hardware. e.g. if access is made to digital I/O, RS485 port or CTNet on SM-Applications Lite module.	Yes	Yes	Yes
76	Module type cannot be resolved. Module is not recognised.	Yes	No	Not Run
77	Inter-Option module comms error with module in slot 1.	Yes	Yes	Yes
78	Inter-Option module comms error with module in slot 2.	Yes	Yes	Yes
79	Inter-Option module comms error with module in slot 3.	Yes	Yes	Yes

**Table 13.5 Application processor error codes (Continued)**

Error Code	Reason	Trip Drive?	ERROR task?	Prog Halted?
80	Inter-Option module comms error with module unknown slot.	Yes	Yes	Yes
81	<ul style="list-style-type: none"> <li>Slot selected as the reference or feedback does not contain a position option module.</li> <li>Attempt to change the Reference source or the Feedback source in more than one task.</li> </ul> APC internal error. See parameter <b>#81.38</b> . This may be caused by one of the following: <ul style="list-style-type: none"> <li>CAM table too small.</li> <li>A change of too many CAM segments has occurred at the CAM table input.</li> <li>CAM is selected but size is zero.</li> <li>CAM absolute mode selected and Reset Index or Reset Position in segment is out of range.</li> </ul>	May	Yes	Yes
82	Communications to drive faulty.	May	Yes	Yes

**Table 13.6 Terminals**

Terminal	Function	Description
1	0V SC	0V connection for EIA-485 port.
2	/RX	EIA-485 Receive line (negative). Incoming.
3	RX	EIA-485 Receive line (positive). Incoming.
4	/TX	EIA-485 Transmit line (negative). Outgoing.
5	TX	EIA-485 Transmit line (positive). Outgoing.
6	CTNet A	CTNet data line.
7	CTNet Shield	Shield connection for CTNet.
8	CTNet B	CTNet data line.
9	0V	0V connection for digital I/O.
10	DIGIN0	Digital input 0.
11	DIGIN1	Digital input 1.
12	DIGOUT0	Digital output 0.
13	DIGOUT1	Digital output 1.

Information
Introduction
Installation
Getting started
SM-Register Functionality
Parameters
DP_L Programming
Communications
Freeze and Marker
CTSync
Inter-option synchronisation
Diagnostics
Quick Reference
Index

---

# Index

---

## A

ANSIREPLY .....	109
AssRAM .....	110

## C

CalculateRegistrationDistance .....	63
CalculateRegistrationPosition .....	62
CModeXfer .....	109
Communications .....	70, 113
ConfigureRegistrationFilter .....	47
ConfigureRegistrationMode .....	45
ConfigureRegistrationWindow .....	49
Connections .....	13, 15
CTNet .....	13, 14, 18
CTNETDIAGNOSTICS .....	108
CTSync .....	124

## D

Diagnostics .....	139
Digital I/O .....	82, 146
DisableRegistrationFilter .....	48
DisableRegistrationMode .....	47
DisableRegistrationWindow .....	51
Downloading .....	22
DPL .....	20, 99, 106

## E

EnableRegistrationFilter .....	48
EnableRegistrationMode .....	47
EnableRegistrationWindow .....	51
Error Codes .....	140

## F

Firmware .....	144
Firmware Version .....	144
Fitting a SM-Register Module .....	11
Freeze .....	122

## G

GETPARATTR .....	108
GetRegistrationEvent .....	51
GetRegistrationFilter .....	58
GetRegistrationFrame .....	53
GetRegistrationPosition .....	64
GetRegistrationStartCondition .....	59
GetRegistrationStatus .....	59
GetRegistrationTrimTimings .....	61
GetRegistrationWindow .....	61

<b>I</b>	
Installation .....	11
Inter-option .....	131
<b>M</b>	
Mapping parameters (fieldbus) .....	120
Marker .....	122, 123
Modbus ASCII .....	119
Modbus-RTU .....	115
Motion Engine .....	126
<b>P</b>	
Parameters .....	67, 104
PFXREAD6/PFXWRITE6 .....	109
PLC Registers .....	79, 104
<b>Q</b>	
Quick Reference .....	145
<b>R</b>	
RamLength .....	110
RS485 .....	15, 18
Run-time Errors .....	139
<b>S</b>	
Safety .....	6
SCHEDULEEVENT .....	108
SetRamLength .....	110
SetRegistrationStartCondition .....	52
Setup Parameters .....	145
SETUSERID .....	109
SMARTCARD .....	11
SyPTPro .....	18, 19
<b>T</b>	
Tasks .....	100
Termination .....	14, 17
Timer .....	80
Timer Unit .....	146
Trip Codes .....	139
<b>U</b>	
UDFB .....	111
UnassRAM .....	110
<b>V</b>	
Variables .....	102
Virtual Master .....	129



**0471-0117-02**