EMERSON
Industrial Automation

User Guide

SI-Applications Plus
SI-Applications Lite V2
SI-Register
Modules

SI-Applications Plus
SI-Applications Lite V2
SI-Register

Part Number: 0478-0009-01
Issue: 1

CONTROL TECHNIQUES

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 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 the guide, without notice.

Drive software version

This product is supplied with the latest software version. If this drive is to be connected to an existing system or machine, all drive software versions should be verified to confirm the same functionality as drives of the same model already present. This may also apply to drives returned from a Control Techniques Service Centre or Repair Centre. If there is any doubt please contact the supplier of the product.

Environmental statement

Control Techniques is committed to minimising the environmental impacts of its manufacturing operations and of its products throughout their life cycle. To this end, we operate an Environmental Management System (EMS) which is certified to the International Standard ISO 14001. Further information on the EMS, our Environmental Policy and other relevant information is available on request, or can be found at www.greendrives.com.

The electronic variable-speed drives manufactured by Control Techniques have the potential to save energy and (through increased machine/process efficiency) reduce raw material consumption and scrap throughout their long working lifetime. In typical applications, these positive environmental effects far outweigh the negative impacts of product manufacture and end-of-life disposal.

Nevertheless, when the products eventually reach the end of their useful life, they must not be discarded but should instead be recycled by a specialist recycler of electronic equipment. Recyclers will find the products easy to dismantle into their major component parts for efficient recycling. Many parts snap together and can be separated without the use of tools, while other parts are secured with conventional fasteners. Virtually all parts of the product are suitable for recycling.

Product packaging is of good quality and can be re-used. Large products are packed in wooden crates, while smaller products come in strong cardboard cartons which themselves have a high recycled fibre content. If not re-used, these containers can be recycled. Polythene, used on the protective film and bags for wrapping product, can be recycled in the same way. Control Techniques' packaging strategy prefers easily-recyclable materials of low environmental impact, and regular reviews identify opportunities for improvement.

When preparing to recycle or dispose of any product or packaging, please observe local legislation and best practice.

REACH legislation

EC Regulation 1907/2006 on the Registration, Evaluation, Authorisation and restriction of Chemicals (REACH) requires the supplier of an article to inform the recipient if it contains more than a specified proportion of any substance which is considered by the European Chemicals Agency (ECHA) to be a Substance of Very High Concern (SVHC) and is therefore listed by them as a candidate for compulsory authorisation.

For current information on how this requirement applies in relation to specific Control Techniques products, please approach your usual contact in the first instance. Control Techniques position statement can be viewed at:

http://www.controltechniques.com/REACH

Issue Number: 1

Firmware: V02.00.00 (Plus, Lite V2, Register)

Contents

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.

NOTE 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/start-up 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 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 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 behavior 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 SAFE TORQUE OFF function 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* 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 Access

Drive access must be restricted to authorized personnel only. Safety regulations which apply at the place of use must be complied with.

1.6 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 (earth) connections.

The drive *User Guide* or *Technical Data Guide* contains instruction 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:

2006/42/EC: Safety of machinery.

2004/108/EC: Electromagnetic Compatibility.

1.7 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 **0.046** motor rated current. This affects the thermal protection of the motor.

1.8 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.

1.9 General safety considerations for remote operation

SI-Applications enables the possibility of remotely controlling a machine from a distance. It is vital that when connecting to a machine remotely, adequate safety procedures are implemented to prevent damage to the machine or injury to personnel.

Any connection to a live system has the possibility of altering the state of the machine, adequate procedures must be implemented to cover this situation. It is the responsibility of the machine builder to ensure that such a system is safe and complies with current legislation.

2 Introduction

2.1 Features for different module variants

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 Second Processors. These Second Processors provide an additional processor for the drive and allow 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 Second Processors are System Integration Modules that can be installed to Option Module slot 3 on Unidrive M. The Second Processor modules are powered from the drive's internal power supply.

Task based programming system allowing for real-time control of drive and process.

Specifications for SI-Applications Plus

- Enhanced high speed dedicated microprocessor
- 384 kb Flash memory for user program
- 80 kb user program memory
- EIA-RS485 port offering ANSI, Modbus-RTU slave and master and Modbus-ASCII slave and master protocols
- CTNet high speed network connection offering up to 5 Mbit/s data rate.
- Two 24 V digital inputs
- Two 24 V digital outputs
- Task based programming system allowing for real-time control of drive and process
- CTSync

Specifications for SI-Applications Lite V2

- Enhanced high speed dedicated microprocessor
- 384 kb executable memory, 80 kb user memory
- Task based programming system allowing for real-time control of drive and process

Specifications for SI-Register

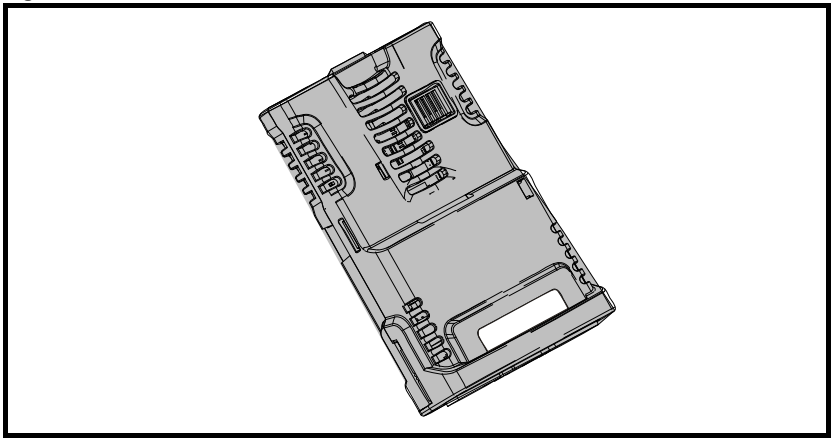
- Registration position capture functionality.
- Enhanced high speed dedicated microprocessor.
- 384kb Flash memory for user program
- 80kb user program memory.
- EIA-485 port offering ANSI, Modbus-RTU slave and master and Modbus-ASCII slave and master protocols.
- CTNet high speed network connection 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.

NOTE SyPTPro V2.6.0 or later is required to use the SI-Applications Plus, SI-Applications LiteV2, or SI-Register.

NOTE Any reference to the SI-Applications Plus module in this User Guide is also applicable to the SI-Register module.

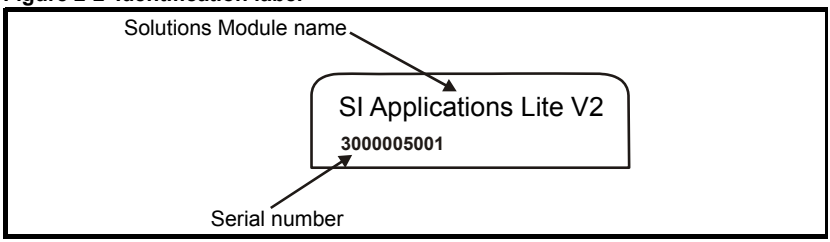
2.2 System Integration Module identification

Figure 2-1 Second Processor



The Second Processors can be identified by the label located on the top of the System Integration Module.

Figure 2-2 Identification label



2.3 Conventions used in this guide

This manual contains information relating to the SI-Applications Lite V2, SI-Applications Plus and SI-Register System Integration Modules.

Throughout the manual, these will be referred to generically as Second Processors or modules. Should a particular section refer to one or two of these modules only, the reference will be to those modules by their name directly. Any reference to the SI-Applications Plus is also applicable to the SI-Register module.

The configuration of the host drive and System Integration Module is done using menus and parameters. A menu is a logical collection of parameters that have similar functionality.

In the case of a System Integration Module, the parameters will appear in menu 15, 16 or 17 for the drive depending on the slot the module is installed into. The SI-Applications Plus, SI-Applications Lite V2 and SI-Register modules can be physically installed into option slot 3 only, with the parameters for the modules appearing in menu 17. The module parameters can be made to appear in menus 15 or 16 by setting Pr **11.056** (Option Slot Identifiers) on the drive.The method used to determine the menu or parameter is as follows:

- Pr **mm.000** - signifies any menu and parameter number 00.
- Pr **mm.PPP** - where **mm** signifies the the menu allocated to the System Integration Module (this could be 15, 16 or 17) and **PPP** signifies the parameter number.

2.4 PC Development Software

Application programs for the Second Processors 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 and option parameters, and program variables.
- Single-stepping and breakpoint debugging facilities.

With SyPTPro you may connect to the Second Processors by either:

- Direct connection to the EIA-RS485 port on the front of the Drive.
- Connecting to one or more options on a CTNet network (a CTNet interface card for the PC will be required). See section *Features* on pages 8, 9 & 10 for availability of CTNet on your Second Processor
- Ethernet.

SyPTPro runs under Microsoft Windows™ 2000 / XP / Vista (32 and 64 bit) and Windows 7 (32 and 64 bit).

2.5 User Knowledge

If developing custom application software it is beneficial to have some understanding of real-time task and event driven programming. A rudimentary understanding of the BASIC programming language is also beneficial but not essential. The ladder diagram (LD) and function block diagram (FBD) facilities of SyPTPro make it much easier for people familiar with PLCs to migrate.

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

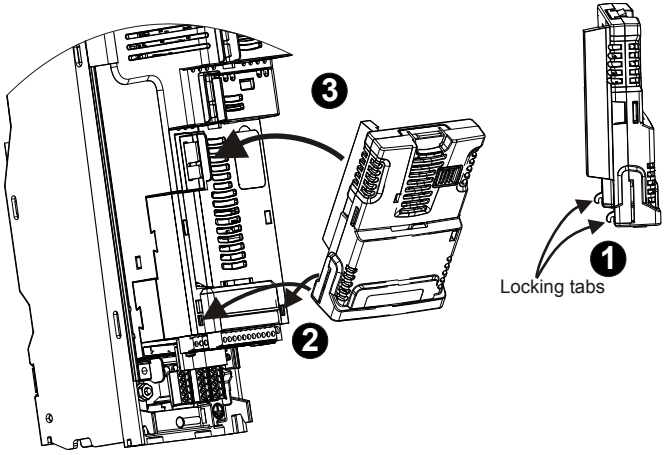
3 Installation

Before installing or removing a System Integration 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 System Integration Module.

3.1 General Installation

The installation of a System Integration Module is illustrated in Figure 3-1.

Figure 3-1 Installing a System Integration Module

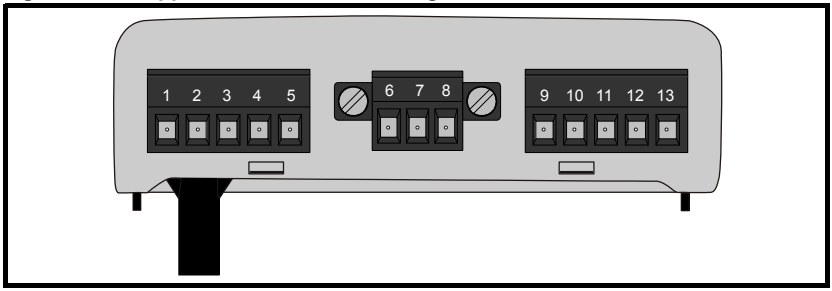


1. Identify the two locking tabs on the rear of the large Option Module
2. Hold the Option Module at a slight angle to the drive, and insert the locking tabs into the slots
3. Push the top of the module into the drive until it locates in the drive body

The System Integration Module connector is located on the underside of the module (1). Push this into the System Integration Module slot located on the drive until it clicks into place (2). For further information, refer to the appropriate drive manual.

3.2 Electrical Connections

Figure 3-2 SI-Applications Plus and SI-Register - Front View



The terminal functions are given in Table 3.1.

Please note that SI-Applications Lite V2 does not have these terminals.

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

Table 3.1 Module Connectors

Terminal	Function	Description
1	0V SC	0V connection for EIA-RS485 port
2	/RX	EIA-RS485 Receive line (negative). Incoming.
3	RX	EIA-RS485 Receive line (positive). Incoming.
4	/TX	EIA-RS485 Transmit line (negative). Outgoing.
5	TX	EIA-RS485 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	DI0	Digital input 0
11	DI1	Digital input 1
12	DO0	Digital output 0
13	DO1	Digital output 1

Table 3-3 Digital input specifications

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

Table 3.2 Digital output specifications

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

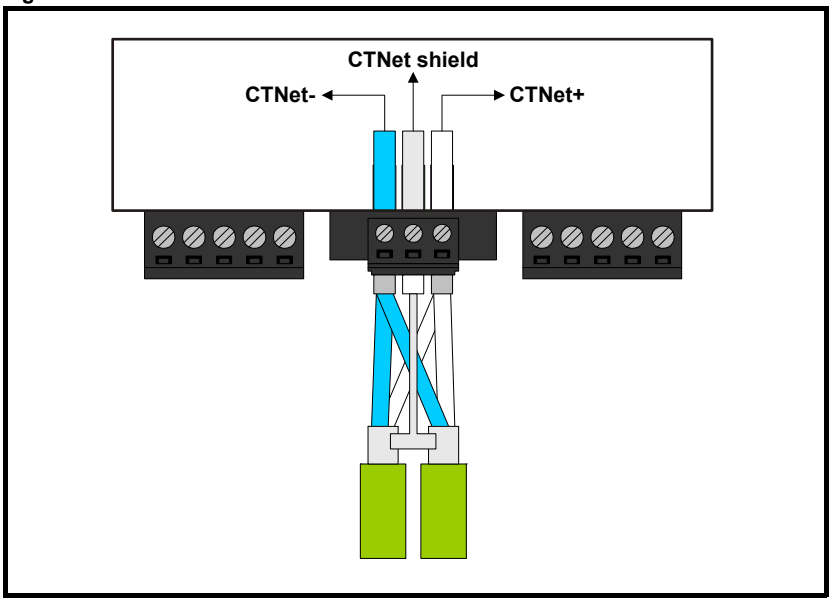
NOTE When inactive they are effectively floating.

3.3 Connections

This User Guide covers only the basics of connecting a CTNet network. Please refer to the CTNet *User Guide* for full information. SI-Applications Lite V2 does not have these CTNet Connections.

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

Figure 3-4 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. This design guarantees shield continuity.

3.4 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.

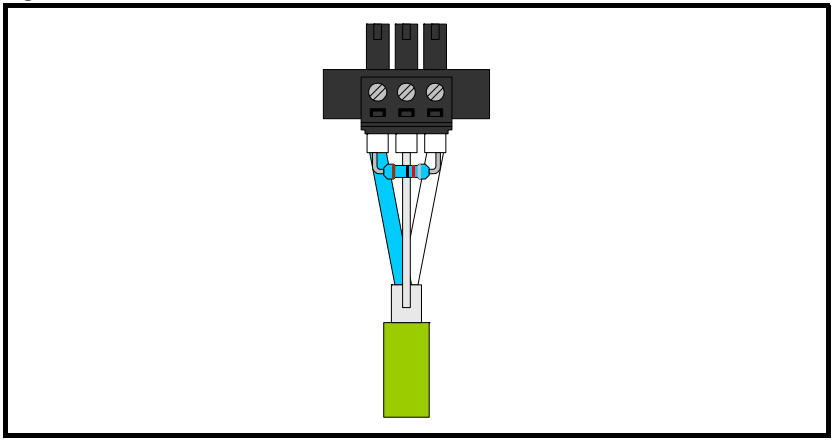
CTNet networks run at high data rates, and require cable specifically designed to carry high frequency signals. Low quality cable will attenuate the signals, and may render the signal unreadable for the other nodes on the network.The only approved CTNet cable is that supplied by Control Techniques.

3.5 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 Ω 0.25 W termination resistor should be installed across the CTNet+ and CTNet- data lines at BOTH ends of the cable run.

Figure 3-5 CTNet Network Termination



WARNING

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. This may cause nodes to miss some bits of information, resulting in transmission errors being reported. If network overload is excessive, the signal levels may be so low that nodes cannot detect any network activity at all.

3.5.1 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 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.

3.5.2 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*.

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 on these points.

3.6 EIA-RS485 Connections

Please note that SI-Applications Lite V2 does not have these connections.

The EIA-RS485 port is available for lower-speed communications (up to 115,200 bit/s). As standard the port supports the CT-ANSI slave, Modbus-RTU master and slave, and Modbus-ASCII master and slave protocols. Both 2 and 4-wire connections are possible.

More information on the use of the EIA-RS485 port can be found in Chapter 6 *Communications* on page 70.

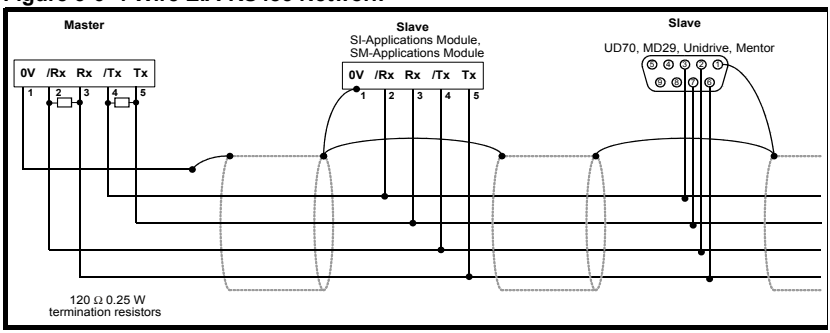
A host controller can operate up to thirty-two EIA RS485 devices with the use of line repeaters. Each transmitter and receiver of Control Techniques devices loads the line by 2 unit loads. Therefore in two-wire mode, each Control Techniques device loads the line by 4 unit-loads. This means that no more 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.

3.6.1 4 Wire EIA-RS485 Network

The diagram below shows the connections required for a 4 wire EIA-RS485 network, using a master controller with an EIA-RS485 port. SI-Applications Modules can be configured to act as master controllers, but this requires DPL programming to control the network.

An EIA-RS232 to EIA-RS485 converter is required to allow a standard PC serial port to communicate with a 4 wire EIA-RS485 network.

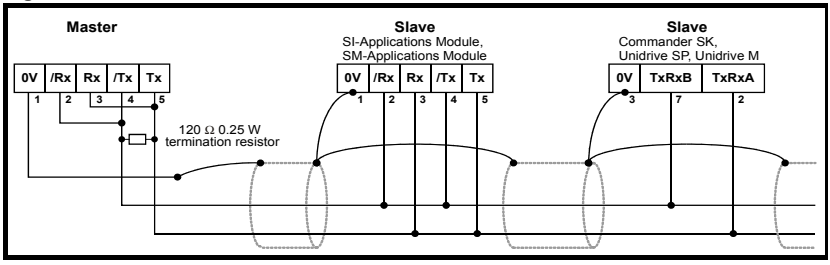
Figure 3-6 4 Wire EIA-RS485 Network



3.6.2 2 Wire EIA-RS485 Network

The diagram below shows the connections required for a 2 wire EIA-RS485 network, using a master controller with an EIA-RS485 port. SI-Applications Modules can be configured to act as master controllers, but this requires DPL programming to control the network.

Figure 3-7 2 Wire EIA-RS485 Network



An EIA-RS232 to EIA-RS485 converter with “intelligent transceiver switching” (also known as “magic” EIA-RS485 converters) is required to allow a standard PC serial port to communicate with a 2 wire EIA-RS485 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.

3.6.3 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.

3.6.4 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 300 mm (1 ft) between the communications cable and the power cable.

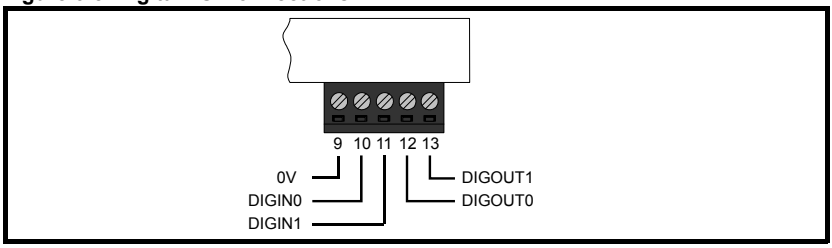
Cables crossing one another at right-angles are unlikely to give trouble. The maximum cable length for a EIA-RS485 jumper (link) is 1200 metres (4,000 ft). This is at low baud rates only. The higher the baud rate the lower the maximum cable length.

3.6.5 Termination

When a long-distance multi-drop EIA-RS485 system is used, the transmit and receive pairs should have a termination resistor of 120 Ω installed across them in order to reduce signal reflections. However, at the lower data rates this is not so critical.

3.7 Digital I/O Connections

Figure 3-8 Digital I/O Connections



The SI-Applications Plus 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/controlled from the user program loaded into the SI-Applications Modules .

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

The digital I/O are controlled using menu 86 - refer to section 5.6 *Menu 86 - Digital I/O Parameters* on page 44.

3.8 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-RS485 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 SI-Applications Modules and some of the aspects of using SyPTPro.

Within the SI-Applications Modules the current slot menu is aliased as menu 81. Therefore when connected to the module via a communications jumper (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 SI-Applications modules with code using menu 81 can be installed to any slot, and the code should run as normal.

When the SI-Applications Modules are installed, the module identification parameter **Pr 81.01** will show the following...

Plus	Lite V2	Register
304	305	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 V02.00.00.

NOTE SI-Applications module specific parameters referred to in this User Guide Menu 70 to 75, 81, 85, 86, 88, 90 and 91 all use the Unidrive SP parameter access mechanism, **MM.PP**, rather than the Unidrive M, **MM.PPP**, access mechanism. This gives legacy compatibility with previously written software

4.1 Using SyPTPro

SyPTPro provides the development platform for the SI-Applications Modules. This is detailed within the SyPTPro Help file.

4.2 Connecting the PC to the Second Processor

There are two methods of connecting the programming PC to the Second Processor 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 3.3 *Connections* on page 14 for details of the CTNet connections on the SI-Applications Modules.

Refer to section 2.1 *Features for different module variants* on page 8 for availability of CTNet.

Plus	Lite V2	Register
✓	×	✓

4.2.2 EIA-RS485 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-RS232 to EIA-RS485 or USB to EIA-RS485 - these leads are used with other Control Techniques products that use a RJ45 EIA-RS485 connector such as the Unidrive SP, Commander SE and Commander SK.

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

Figure 4-1 Communications Cable

A number of drives may be connected via their EIA-RS485 ports on an RTU network, thus allowing the user to control any of those drives. In addition, if a drive has an SI-Applications Modules installed these will be seen by SyPT Pro along with any other SI-Applications Modules connected to this over CTNet. This is part of the routing capability of the CTNetAPI. See section 4.3 *CTNetAPI routing* on page 20.

Plus	Lite V2	Register
✓	×	✓

4.3 CTNetAPI routing

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

Plus	Lite V2	Register
✓	×	✓

4.4 Configuring Communications within SyPTPro

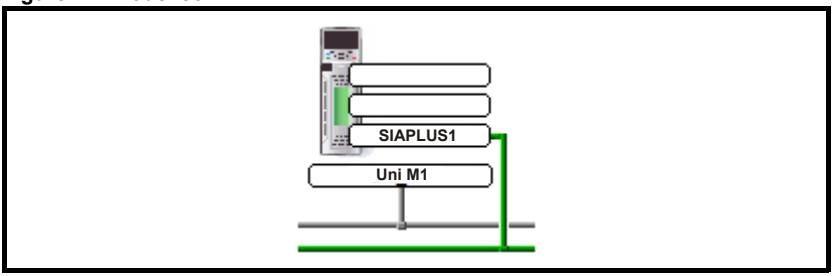
Before attempting to go *on-line* to the SI-Applications Modules you must set SyPTPro up to use the correct communications protocol:

1. In the SyPTPro Configuration Editor select **PC Communications Settings** from the Run menu
2. If connecting via CTNet (SI-Applications only), 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 parameter Pr **11.025** is set to **19200** (this is the default value).
4. Press OK.

SI-Applications Plus	SI-Applications Lite V2	SI-Register
✓	×	✓

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 System Integration Modules can be entered using the tabs at the top of the box.
3. Press OK.

4.6 DPL Programming Basics

The SI-Applications Modules 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*.

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 8 ms or when the module first powers-up. Each task has a particular name, purpose and priority. Refer to section 7.2 *Tasks* on page 80 for further information.

4.6.1 Function Block Library

SyPTPro comes 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.

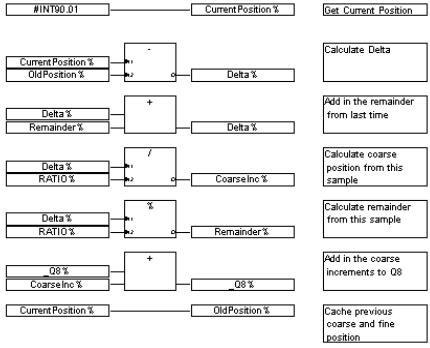
You can also create your own function blocks within your program. So if you've created a new profile generator, you may encapsulate it within a user-defined function block (UDFB) and incorporate into your main DPL program. See section 7.7 *User Defined Function Blocks* on page 92 and the on-line help for information.

4.7 Program Example

Figure 4-3 is an example of a DPL program written within SyPTPro:

Figure 4-3 Example DPL Program

```
$AUTHOR   Kevin Vedmore
$COMPANY  CT SSPD
$TITLE    Q8 Counter
$VERSION  V1.0
$DRIVE    UNIDRVSP

$DEFINE RATIO% 65536
 Initial(

#81.12=3                    //1ms POS task
REINIT                      //Reinitialise (ensure that new setup is active)

Delta% = 0                  //Initialise Delta
CoarseInc% = 0              //Coarse Position
Remainder% = 0              //Fine Position
CurrentPosition% = 0        //Current Position
OldPosition% = 0            //Previous Position

) //Initial

 Pos0(
```




```
) //Pos0
```

This program will take the positional feedback information from the drive (which is scaled to 2^{32}/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 to Unidrive M as 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.

4.7.2 Initial Task

As explained later in section 7.2 *Tasks* on page 80, this is a *task* which is executed when the SI-Applications Modules 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 initialize 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 5.4 *Menus 70-75 - PLC Registers* on page 40.

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 1 ms
REINIT     //Reinitialize
```

4.8 Downloading Programs

By default, programs can only be downloaded to the SI-Applications Modules when the drive enable signal is not active (**Pr 06.015**=0). This behavior can be disabled by setting **Pr 81.37** to 0.

5 Parameters

5.1 Overview

The Second Processor contains two parameter databases:

- The drive database
  This contains the entire drive parameter set. The Second Processor 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 "Waiting For Options" will appear for a few seconds on the drive display. This will not occur again unless the Second Processor is moved to a different drive with different firmware or the drive firmware is updated.
- The Second Processor 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)

5.2 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:

5.2.1 Saving Second Processor Parameters

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

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

NOTE Menu 90 and 91 parameters are not persistent over a reset or power cycle and cannot be saved.

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.

5.2.2 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 5.2.3 *Saving Drive Parameters*.

5.2.3 Saving Drive Parameters

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

- Menus 1 through 14, 18, 19, 21, 22, 23 and 29 through 41.
- Menus 15, 16 and 17, if a module is present in the relevant slot.

To save the drive parameters:

1. Set Pr **mm.000**=1000 (parameter zero in any menu when using drive keypad)
2. Set Pr **10.038**=100 (simulates pressing the reset button on the drive keypad)

NOTE A drive save can be performed by entering a value of 1000 in any Pr **mm.000** menu and pressing the reset button. For drives that support a 24 V supply 1001 should be used while running on 24 V.

5.3 Configuration Parameters

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

Slot	Menu
1	15
2	16
3	17

A Second Processor module can be physically installed into option slot 3 only, with the parameters for the module appearing in menu 17. However, the module parameters can be made to appear in menus 15 or 16 by setting Pr **11.056** (Option Slot Identifiers) on the drive.

In addition to these menus, an alias of the appropriate menu is available as local menu 81 within the Second Processor. This menu can be accessed from the user DPL program or via communications (CTNet/CT-RTU/EIA-RS485) and provides a convenient way to read or change the setup parameters without having to know which slot the Second Processor is installed in.

WARNING Unless otherwise indicated, these parameters are only read when the Second Processor 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 Second Processor 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. "Initialization" means that the parameter is read only on module reset or REINIT DPL command.

TIP 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, but menus 15 to 20 and 24 to 28 will be left unchanged.

5.3.1 Parameter Descriptions

Pr 81.01	Module Code		
Access	RO	Range	0 to 499
Default	N/A	Update Rate	N/A

Plus	Lite V2	Register
304	305	306

Pr 81.02	Firmware Version		
Access	RO	Range	00.00 to 99.99.99
Default	N/A	Update Rate	N/A

Plus	Lite V2	Register
✓	✓	✓

Specifies the major revision number of the operating system of the Second Processor. 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	1 ms of change

Provides the run status of the user DPL program in the Second Processor. 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

Plus	Lite V2	Register
✓	✓	✓

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

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

Plus	Lite V2	Register
✓	✓	✓

Pr 81.05	EIA-RS485 Address		
Access	RW	Range	0 to 255
Default	11	Update Rate	Initialization

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-RS485 mode is 25 (CTSync) or 26 (CTSync).

Plus	Lite V2	Register
✓	×	✓

Pr 81.06	EIA-RS485 Mode		
Access	RW	Range	0 to 255
Default	1	Update Rate	Initialization

Defines the mode of operation (or protocol) for the on-board EIA-RS485 port. For details of these modes, refer to Table 5.1 *Serial modes - Pr 81.06*

Table 5.1 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-RS485 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-RS485 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

Table 5.1 Serial modes - Pr 81.06

Mode	Description
15, 45, 75	2-wire Modbus RTU slave The EIA-RS485 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-RS485 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-RS485 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-RS485 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
19, 49, 79	2-wire Modbus RTU master The EIA-RS485 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-RS485 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	SI-Applications Modules CT-Sync Master The baud rate is fixed at 896875bps
26	SI-Applications Modules CT-Sync Slave The baud rate is fixed at 896875 bps

Plus	Lite V2	Register
✓	×	✓

Pr 81.07	EIA-RS485 Baud Rate		
Access	RW	Range	0-9 (300-115200 bps)
Default	4 (4800)	Update Rate	Initialization

Defines the baud-rate (or bits-per-second) for the on-board EIA-RS485 port. The following are supported:

NOTE

This parameter is not relevant when the EIA-RS485 port mode is set to 25 (CTSync Master) or 26 (CTSync Slave).

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

Plus	Lite V2	Register
✓	×	✓

Pr 81.08	EIA-RS485 Turn-around Delay		
Access	RW	Range	0 to 255 ms
Default	2 ms	Update Rate	Initialization

Defines a fixed delay between receiving a message on the EIA-RS485 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 1 ms delay and this parameter can be used to extend it.

Plus	Lite V2	Register
✓	×	✓

Pr 81.09	EIA-RS485 Tx Enable Delay		
Access	RW	Range	0 to 1 ms
Default	0 ms	Update Rate	Initialization

This parameter allows a 1 ms delay to be introduced between the Second Processor enabling the EIA-RS485 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.

Plus	Lite V2	Register
✓	×	✓

Pr 81.10	DPL Print Routing		
Access	RW	**Range**	0/1
Default	0	**Update Rate**	Initialization

This parameter is available on SI-Applications Lite V2 module but is not allowed to be set to a 1 (On). 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-RS485 port.

Plus	Lite V2	Register
0=✓ 1=✓	0=✓ 1=×	0=✓ 1=✓

Pr 81.11	Clock Task Scheduling (ms)		
Access	RW	**Range**	0 to 200 ms
Default	10 ms	**Update Rate**	Initialization

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

Plus	Lite V2	Register
✓	✓	✓

Pr 81.12	POS task scheduling rate		
Access	RW	**Range**	0 to 6
Default	0	**Update Rate**	Initialization

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	250 µs
0.5	2	500 µs
1	3	1 ms
2	4	2 ms
4	5	4 ms
8	6	8 ms

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.

Plus	Lite V2	Register
✓	✓	✓

Pr 81.13	Auto-run Enable		
Access	RW	Range	0/1
Default	1	Update Rate	Initialization

Plus	Lite V2	Register
✓	✓	✓

Pr 81.14	Global Run-time Trip Enable		
Access	RW	Range	0/1
Default	0	Update Rate	Initialization

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

For more information, see section 12.1 *Run-time errors* on page 155.

Plus	Lite V2	Register
✓	✓	✓

Pr 81.15	Disable Reset on Trip Cleared		
Access	RW	Range	0/1
Default	0	Update Rate	Initialization

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).

Plus	Lite V2	Register
✓	✓	✓

Pr 81.16	Encoder Data Update Rate		
Access	RW	Range	0-3
Default	0	Update Rate	Initialization

Display	Description
0	APC data and menu 90 encoder parameters are updated every 250 µs.
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.

Plus	Lite V2	Register
✓	✓	✓

Pr 81.17	Enable Parameter Over-range Trips		
Access	RW	**Range**	0/1
Default	0	**Update Rate**	Initialization

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.

Plus	Lite V2	Register
✓	✓	✓

Pr 81.18	Watchdog Enable		
Access	RW	**Range**	0/1
Default	0	**Update Rate**	Initialization

When set it enables the DPL program watchdog function. The DPL WDOG command must then be executed every 200 ms. This can be used to protect the program against malfunction. If the command is not executed within a 200 ms time period the drive will trip on **Slotx Watchdog**. 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.

Plus	Lite V2	Register
✓	✓	✓

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

Setting this parameter to 1 will initiate a save of all non-volatile Second Processor data. This save may take up to approximately 100 ms before it is actioned. This consists of the P/Q/T/U PLC register sets and optionally menu 20 (depending upon the setting of Pr **81.21**).

Plus	Lite V2	Register
✓	✓	✓

NOTE This will also cause a reset of the module and this parameter will revert to zero automatically. Also if the drive is tripped it will be reset. Menu 81 will not be saved.

Pr 81.20	Enable “UU trip” Save		
Access	RW	**Range**	0/1
Default	0	**Update Rate**	Immediate

Plus	Lite V2	Register
✓	✓	✓

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

NOTE Note that when a ‘UU’ save occurs the Second Processor will be reset.

Plus	Lite V2	Register
✓	✓	✓

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

If set to 1, menu 20 will be saved/restored along with other non-volatile parameters upon a save request (Pr **xx.019**=1) or power-down save (Pr **xx.020**=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 Second Processor is installed to the drive **only one** should have this parameter set otherwise menu 20 will not be restored correctly on power-up.

Plus	Lite V2	Register
✓	✓	✓

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

Pr 81.22	CTNet Token Ring ID		
Access	RW	Range	0 to 255
Default	0	Update Rate	Initialization

This parameter allows the user to specify the identity of the CTNet token ring to which a Second Processor 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.

Plus	Lite V2	Register
✓	×	✓

Pr 81.23	CTNet Node Address		
Access	RW	Range	0 to 255
Default	0	Update Rate	Initialization

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.

Plus	Lite V2	Register
✓	×	✓

Pr 81.24	CTNet Baud Rate		
Access	RW	Range	0 to 3
Default	1(2.5)	Update Rate	Initialization

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	5 Mbs	1.250	2	1.25 Mbs
2.500	1	2.5 Mbs	0.625	3	625 kbs

Plus	Lite V2	Register
✓	×	✓

Pr 81.25	CTNet Sync Setup		
Access	RW	Range	0 to 9999
Default	0	Update Rate	Initialization

Specifies the synchronization 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 150 ms. When using easy mode (see below) it is only necessary to set up the FF (fast cyclic rate).

Plus	Lite V2	Register
✓	×	✓

Pr 81.26, Pr 81.28, Pr 81.30	CTNet Easy Mode Setup Parameters		
Access	RW	Range	0 to 25503
Default	0	Update Rate	Initialization

Plus	Lite V2	Register
✓	×	✓

Pr 81.27, Pr 81.29, Pr 81.31 - Pr 81.34	CTNet Easy Mode Setup Parameters		
Access	RW	Range	0 to 9999
Default	0	Update Rate	Initialization

NOTE Easy mode cyclic link parameter source and destinations do not accept the **MM.PPP** style mappings for Unidrive M, only **MM.PP**. Where Pr **1.021** is to be accessed remove the first 0 after the decimal point to give a reference of 121, where 121 could be entered in Pr **81.27**, Pr **81.29**, Pr **81.31** to Pr **81.34**.

Plus	Lite V2	Register
✓	×	✓

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

Parameter	Format	Channel	Description
Pr **81.26**	NNNSS	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 **Pr 3.002** (speed)
Pr **81.28**	NNNSS	2	Destination node number and slot for channel 2
Pr **81.29**	MMPP	2	Source drive parameter for channel 2
Pr **81.30**	NNNSS	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

Pr 81.35	CTNet Sync. Event Task ID		
Access	RW	**Range**	0 to 4
Default	0	**Update Rate**	Initialization

Identifies which of the EVENT tasks will be scheduled when a CTNet synchronization message is received or generated. Synchronization 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			

Plus	Lite V2	Register
✓	×	✓

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

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

Pr 81.36	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	Initialization Error	The SI-Applications Modules module was unable to configure the CTNet interface. Check that the node address and data rate are set correctly.
-3	MYRECON	The SI-Applications Modules module forced a CTNet network reconfiguration

Plus	Lite V2	Register
✓	×	✓

Pr 81.37	Reject Download if Drive Enabled		
Access	RW	Range	0/1
Default	0	Update Rate	Initialization

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.

Plus	Lite V2	Register
✓	✓	✓

Pr 81.38	APC Run-time trip		
Access	RW	Range	0/1
Default	0	Update Rate	Initialization

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 unitialized CAM function. When this parameter is 1 the drive will not trip when an APC non-recoverable error occurs.

Plus	Lite V2	Register
✓	×	✓

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 synchronization status.

Synchronization Status	Status
0	The synchronization master request is zero or another System Integration Module is synchronization master.
1	The System Integration Module is synchronization master.
3	The System Integration Module is synchronization master, but the synchronization frequency is out of specification or not present.

Plus	Lite V2	Register
✓	✓	✓

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

Plus	Lite V2	Register
✓	✓	✓

Pr 81.42	Pass Freeze Through to Drive		
Access	RW	Range	0/1
Default	0	Update Rate	Initialization

When this parameter is ON (1), the voltage on the Second Processor digital input 0 (zero) is passed through to the drive's internal Freeze line. This can be seen by other classes of SI-System Integration Modules. For further information on the Freeze Input refer to Chapter 8 *Freeze and marker* on page 94. When Pr **81.42** is set to 1, Pr **03.100** on the drive is set to 4. This sets the common freeze line as the F1 freeze trigger source on the drive.

Plus	Lite V2	Register
✓	×	✓

Pr 81.43	Freeze Invert		
Access	RW	Range	0/1
Default	0	Update Rate	Initialization

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 8 *Freeze and marker* on page 94.

Plus	Lite V2	Register
✓	×	✓

Pr 81.44	Task Priority Level		
Access	RW	Range	0 to 255
Default	0	Update Rate	Initialization

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	

Plus	Lite V2	Register
✓	✓	✓

Pr 81.45	User Set-Up Parameter 1		
Access	RO	Range	N/A
Default	0	Update Rate	N/A

This parameter is dependant on what is running in the module i.e. Indexer.

Plus	Lite V2	Register
✓	✓	✓

Pr 81.46	User Set-Up Parameter 2		
Access	RO	Range	N/A
Default	0	Update Rate	N/A

This parameter is dependant on what is running in the module i.e. Indexer.

Plus	Lite V2	Register
✓	✓	✓

Pr 81.47	User Set-Up Parameter 3		
Access	RO	Range	N/A
Default	0	Update Rate	N/A

This parameter is dependant on what is running in the module i.e. Indexer

Plus	Lite V2	Register
✓	✓	✓

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).

Plus	Lite V2	Register
✓	✓	✓

Pr 81.49	User program ID		
Access	RO/RW	Range	Signed 16-bit
Default	0	Update Rate	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.

Plus	Lite V2	Register
✓	✓	✓

Pr 81.50	Run-time Error Code		
Access	RO	Range	0 to 255
Default	0	Update Rate	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 155 for further information.

Plus	Lite V2	Register
✓	✓	✓

Pr 81.51	Firmware - Minor Version		
Access	RO	Range	0 to 99
Default	N/A	Update Rate	N/A

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

Plus	Lite V2	Register
✓	✓	✓

5.4 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]		
72	RW	_Rx%, _R%[x]		Used for outgoing CTNet cyclic data links. Non-saveable.
73	RW	_Sx%, _S%[x]		Used to incoming CTNet cyclic data links. Non-saveable
74	RW	_Tx%, _T%[x]		General purpose. Saveable.
75	RW	_Ux%, _U%[x]		

You will see from the above table that each parameter within menus 70 to 75 has and 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 5.2 *Saving Parameters* on page 24 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.

Pr **72.79** and Pr **73.79** will be reserved if AUTOSYNC is in use.

Parameters Pr **71.80** thru Pr **71.99** may be used for the RAM File Recording

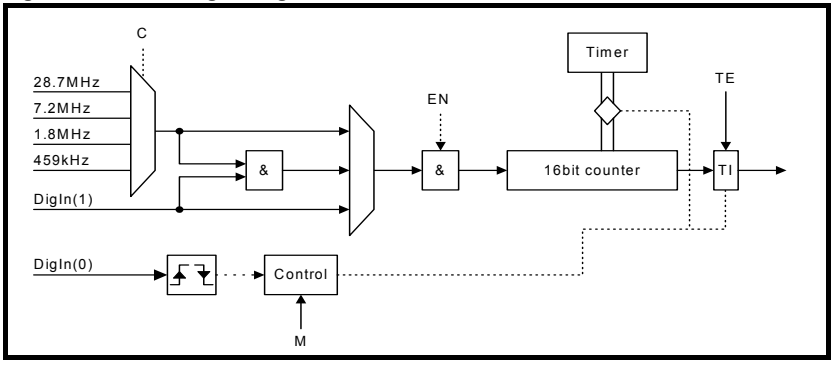
Plus	Lite V2	Register
✓	✓	✓

5.5 Menu 85 - Timer Function Parameters

A hardware/counter is built into the Second Processor which has the following features:

- A 16 bit incremental counter.
- Count rate selectable from the internal clock. The clock rate divisor is selectable from rate/ 1, rate/ 4, rate/ 16, rate/ 64.
- Count rate selectable from an external clock via the DIGIN1 digital input. The maximum clock rate is 600 kHz.
- 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 5-1 Timer Logic Diagram



Plus	Lite V2	Register
✓	×	✓

Pr 85.01	Timer Unit Control Word		
Access	RW	Range	13 bit
Default	N/A	Update Rate	Immediate

Table 5.2 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 (28.7 MHz) 1=rate /4 2=rate /16 3=rate /64
b7-b8	M	Timer mode: 0=Free Running mode The selected clock drives the counter. The TI flag is set on wrap-around. 1=Capture mode 1 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). 2=Capture mode 2 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).

Plus	Lite V2	Register
✓	×	✓

Pr 85.02	Timer Unit Status Word		
Access	RO	Range	0 to 3
Default	N/A	Update Rate	Immediate

Table 5.3 Status Word - Pr 85.02

Bit	Symbol	Function
b0	TI	Timer event flag: 0=No event has occurred 1=An event has occurred (see description for Pr **85.01**) **Note:** 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.
b1	OV	Wrap-around flag 0=Wrap-around has not occurred 1=Counter wrap-around has occurred **Note:** This flag is valid for ALL timer modes and is automatically cleared when the status register is read.

Plus	Lite V2	Register
✓	×	✓

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.

Plus	Lite V2	Register
✓	×	✓

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.

Plus	Lite V2	Register
✓	×	✓

Setting a small wrap-around figure with the timer operating at fast rates, could result in the Second Processor 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 Second Processor 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 **81.13**=0). After the reset the user can reinitialize the timer with a more reasonable wrap-around number.

Pr 85.05	Timer Unit Timer Capture Cache		
Access	RW	Range	16 bit
Default	N/A	Update Rate	Immediate

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

Plus	Lite V2	Register
✓	×	✓

5.6 Menu 86 - Digital I/O Parameters

The SI-Applications Plus and SI-Register have two digital outputs and two digital inputs on-board. These outputs and inputs are controlled though this menu. See Chapter 3 *Installation* on page 12 for detailed specifications.

Pr 86.01	Digital Input 0		
Access	RO	Range	0/1
Default	N/A	Update Rate	Immediate

Plus	Lite V2	Register
✓	×	✓

Pr 86.02	Digital Input 1		
Access	RO	Range	0/1
Default	N/A	Update Rate	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.

Plus	Lite V2	Register
✓	×	✓

Pr 86.03	Digital Output 0		
Access	RW	Range	0/1
Default	N/A	Update Rate	Immediate

Plus	Lite V2	Register
✓	×	✓

Pr 86.04	Digital Output 1		
Access	RW	Range	0/1
Default	N/A	Update Rate	Immediate

Plus	Lite V2	Register
✓	×	✓

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 (+24 V).

Pr 86.05	Digital Outputs 0 and 1		
Access	RW	**Range**	0 to 3
Default	N/A	**Update Rate**	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**).

Plus	Lite V2	Register
✓	×	✓

5.7 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 Second Processor 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.

Plus	Lite V2	Register
✓	✓	✓

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 155.

Plus	Lite V2	Register
✓	✓	✓

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.

Plus	Lite V2	Register
✓	✓	✓

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.

Plus	Lite V2	Register
✓	✓	✓

Pr 88.05	Motion Engine Peak Resource Detect		
Access	RW	Range	0 to 95
Default	0	Update Rate	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.

Plus	Lite V2	Register
✓	✓	✓

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, Pr **88.07** and Pr **88.08** become active. If set to zero, Pr **88.07** and Pr **88.08** will read zero.

Plus	Lite V2	Register
✓	✓	✓

Pr 88.07	Free Resource for Clock Task		
Access	RO	Range	0 to 95
Default	NA	Update Rate	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.

Plus	Lite V2	Register
✓	✓	✓

Pr 88.08	Clock Task Peak Resource Detect		
Access	RO	Range	0 to 95
Default	NA	Update Rate	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.

Plus	Lite V2	Register
✓	✓	✓

5.8 Menu 89 - SI-Applications Plus RS485 Mode 4 Cascade mode

This mode is a legacy mode provided to support older projects with migration to the SI-Applications range of modules, and is largely superseded by CT Net. However, where there is no viable alternative this mode can be used.

The transmitting unit and/or receiving unit can be another MD29, a UD70, or a Mentor II drive. This mode is typically used in wire-drawing type applications - the first drive is given a speed reference from an external source (e.g. a potentiometer). Mode 4 communications are used to pass that reference down to the next drive on the line which uses an Applications Module to receive the data and apply draw. That drive then passes the new reference down to the next drive, and so on.

The data format is 4800 baud, 8 data bits and 1 stop bit with even parity indicating high-order byte and odd parity indicating low-order byte.

Message length is 2-bytes.

Data is transmitted 1 ms after data has been successfully received, in a half-duplex fashion.

Note that if no data is received for 14 ms (slightly longer than a worst-case "normal" message receipt period) a single transmission will be actioned and the module will switch back to receive mode. No trip will be generated. This prevents the module from becoming inactive under communications loss conditions.

The low-order byte is transmitted 2 ms after the high-order byte. This means that the receive-transmit cycle time of SI-Applications Plus in this mode is in the order of 12 ms; 6 ms for the incoming data to be received followed by 6 ms to transmit before switching back to receive mode.

No scaling is applied - it is intended that the user apply any scaling between the values in Pr **89.01** and Pr **89.02**.

As data is transmitted as soon as possible it is likely that the transmission will be made before the user has had time to read the received value and recalculate a new value to be transmitted. This will lead to a ripple delay of 12 ms per module.

Pr 89.01	Received Data		
Access	RO	**Range**	16 bit
Default	N/A	**Update Rate**	N/A

This parameter contains the data received from the transmitting drive.

Pr 89.02	Transmitted Data		
Access	RW	**Range**	16 bit
Default	N/A	**Update Rate**	N/A

This parameter contains the data to be transmitted by the transmitting drive.

5.9 Menu 90 - General Parameters

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

Pr 90.01	Feedback Encoder Position (2^{32}/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.

Plus	Lite V2	Register
✓	✓	✓

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.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.03	Reference Encoder Position (2^{32}/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.

Plus	Lite V2	Register
✓	✓	✓

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.

Plus	Lite V2	Register
✓	✓	✓

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 **11.031** or Pr **00.048** 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.

Plus	Lite V2	Register
✓	✓	✓

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 5.4 Control Word

Bit	Description
b15	If set, the value of Pr **01.046** is set from b6
b14	If set, the value of Pr **01.045** is set from b5
b13	Sets the value of Pr **18.033** (Application menu 1, bit 3)
b12	If set, the value of Pr **06.032** is set from b3
b11	If set, the value of Pr **06.031** is set from b2
b10	If set, the value of Pr **06.030** is set from b1
b9	If set, the value of Pr **06.015** is set from b0
b8	Sets the value of Pr **18.032** (Application menu 1, bit 2)
b7	Sets the value of Pr **18.031** (Application menu 1, bit 1)
b6	Sets the value of Pr **01.046** (Preset select bit 1)
b5	Sets the value of Pr **01.045** (Preset select bit 0)
b4	User trip. Trips drive immediately if set.
b3	Sets the value of Pr **06.032** (Sequencing bit 2: Run Reverse)
b2	Sets the value of Pr **06.031** (Sequencing bit 1: Jog)
b1	Sets the value of Pr **06.030** (Sequencing bit 0: Run forward)
b0	Sets the value of Pr **06.015** (Drive enable)

Table 5.5 Status Word

Bit	Description
b15	Not used
b14	Pr **10.15** (Mains Loss)
b13	Pr **10.14** (Direction running)
b12	Pr **10.13** (Direction commanded)
b11	Pr **10.12** (Braking resistor alarm)
b10	Pr **10.11** (Braking IGBT active)
b9	Pr **10.10** (Regenerating)
b8	Pr **10.09** (Drive output is at current limit)
b7	Pr **10.08** (Load reached)
b6	Pr **10.07** (Above set speed)
b5	Pr **10.06** (At speed)
b4	Pr **10.05** (Below set speed)
b3	Pr **10.04** (Running at or below min speed)
b2	Pr **10.03** (Zero speed)
b1	Pr **10.02** (Drive running)
b0	Pr **10.01** (Drive OK)

Plus	Lite V2	Register
✓	✓	✓

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

For a description see below.

Plus	Lite V2	Register
✓	✓	✓

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

For description see below.

Plus	Lite V2	Register
✓	✓	✓

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

For description see below.

Plus	Lite V2	Register
✓	✓	✓

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

For description see below.

Plus	Lite V2	Register
✓	✓	✓

NOT CURRENTLY SUPPORTED

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 System Integration Module initiated 32 = CTNet 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	250 µs

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 reacitvate it simply set it to zero.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.19	Feedback Encoder Freeze Position		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	250 µs

For description see Pr **90.20**

Plus	Lite V2	Register
✓	✓	✓

Pr 90.20	Feedback Encoder Freeze Turns		
Access	RO	**Range**	Unsigned 16-bit
Default	N/A	**Update Rate**	250 µs

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

Plus	Lite V2	Register
✓	✓	✓

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.

Plus	Lite V2	Register
✓	✓	✓

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 to the drive via serial comms with the drive. Pr **03.026** determines which of the P1 and P2 interfaces on the drive the module will communicate with.

Plus	Lite V2	Register
✓	✓	✓

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 to the drive via serial comms with the drive. Pr **03.026** determines which of the P1 and P2 interfaces on the drive the module will communicate with.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.24	Module slot number		
Access	RO	Range	Unsigned 8-bit
Default	N/A	Update Rate	Initialization

This parameter reports the slot number into which the module is installed. This parameter will reflect any changes made to the option slot menu assignments in Pr **11.056** on the drive.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.25	Feedback encoder marker position (2^{32}/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.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.26	Feedback encoder marker turns (2^{16}/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.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.27	Second Processor database version number		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	Initialization

The database version number is read from the database after power-up.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.28	Reference Encoder Freeze flag		
Access	RW	Range	0/1
Default	N/A	Update Rate	250 µs

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 reacitvate it simply set it to zero.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.29	Reference Encoder Freeze Position		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	250 µs

See Pr **90.30** for description.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.30	Reference Encoder Freeze Turns		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	250 µs

These 2 parameters store the position and turns respectively of the reference encoder at the time the freeze input was activated.

Plus	Lite V2	Register
✓	✓	✓

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.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.32	Reference Encoder Turns and Coarse Position		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	See Pr **81.16**

Plus	Lite V2	Register
✓	✓	✓

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	250 µs

See Pr **90.34** for description.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.34	Reference Encoder Freeze Turns and Coarse Position		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	250 µs

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.

Plus	Lite V2	Register
✓	✓	✓

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.

Plus	Lite V2	Register
✓	✓	✓

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.

Plus	Lite V2	Register
✓	✓	✓

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.

Plus	Lite V2	Register
✓	✓	✓

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.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.39	Drive Keypad Button Status		
Access	RO	**Range**	Signed 16-bit
Default	N/A	**Update Rate**	> 40 ms

The state of the Reverse, Run and Stop keys can be read using this parameter. The keys are represented by bits as follows:

Value	Description
b0	Rev
b1	Run
b2	Stop

Plus	Lite V2	Register
✓	✓	✓

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 Second Processor Event tasks.

Value	Action
0	Do not trigger Event task
1	Trigger Event task
2	Trigger Event1 task
3	Trigger Event2 task
4	Trigger Event3 task

Plus	Lite V2	Register
✓	✓	✓

Pr 90.41	Reference Encoder Marker Flag		
Access	RW	**Range**	0/1
Default	N/A	**Update Rate**	See Pr **81.16**

See Pr **90.42** for description.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.42	Feedback Encoder Marker Flag		
Access	RW	**Range**	0/1
Default	N/A	**Update Rate**	See Pr **81.16**

Plus	Lite V2	Register
✓	✓	✓

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	P1 Drive encoder
1	P1 Slot 1
2	P1 Slot 2
3	P1 Slot 3
4	User program
5	Unconfigured
6	P2 Drive encoder
7	P2 Slot 1
8	P2 Slot 2
9	P2 Slot 3
10	P1 Slot 4
11	P2 Slot 4

Plus	Lite V2	Register
✓	✓	✓

Pr 90.45	Reference Marker Flag Enable		
Access	RW	Range	0/1
Default	N/A	Update Rate	Immediate

See Pr **90.46** for description.

Plus	Lite V2	Register
✓	✓	✓

Pr 90.46	Feedback Marker Flag Enable		
Access	RW	Range	0/1
Default	N/A	Update Rate	Immediate

Plus	Lite V2	Register
✓	✓	✓

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.

Pr 90.47	Reference Freeze Enable		
Access	RW	Range	0/1
Default	N/A	Update Rate	Immediate

Plus	Lite V2	Register
✓	✓	✓

Pr 90.48	Feedback Freeze Enable		
Access	RW	Range	0/1
Default	N/A	Update Rate	Immediate

Plus	Lite V2	Register
✓	✓	✓

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.

Pr 90.49	APC Runtime Error ID		
Access	RO	**Range**	32bit
Default	N/A	**Update Rate**	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*.

Value	Description
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

Plus	Lite V2	Register
✓	✓	✓

5.10 Menu 91 - Fast Access Parameters

The parameters in this menu are Second Processor virtual parameters which provide a faster update rate or enhanced resolution than drive parameters.

Pr 91.01	Short-cut enable		
Access	RW	**Range**	Unsigned 8-bit
Default	0	**Update Rate**	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.

Bit	Function	Related Parameter
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	

Plus	Lite V2	Register
✓	✓	✓

Pr 91.02	Speed set-point (Pr 01.021)		
Access	RW	**Range**	Signed 32-bit
Default	N/A	**Update Rate**	250 µs

Sets the speed reference in units of **0.001 rpm**. This value is mirrored in drive Pr **01.021** (preset speed 1), therefore in order to control the drive speed with this parameter ensure preset speed 1 is selected on the drive (Pr **01.014**=3, Pr **01.015**=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.

Plus	Lite V2	Register
✓	✓	✓

Pr 91.03	Hard-speed reference (Pr 03.022)		
Access	RW	**Range**	Signed 32-bit
Default	N/A	**Update Rate**	250 µs

Controls the hard-speed reference on the drive in units of **0.001 rpm**.

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

This parameter is valid only in Closed Loop Vector and Servo modes only.

Plus	Lite V2	Register
✓	✓	✓

Pr 91.04	Torque setpoint (Pr 04.008)		
Access	RW	**Range**	Signed 32-bit
Default	N/A	**Update Rate**	250 µs

Specifies the torque setpoint (drive Pr **04.008**) in units of 0.01 %.

Ensure bit 2 of Pr **91.01** is set in order to use this parameter.

Plus	Lite V2	Register
✓	✓	✓

Pr 91.05	Full scale speed (rpm)		
Access	RW	**Range**	Signed 32-bit
Default	1500	**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 over range run-time error.

When the drive is in Closed Loop Vector mode, in the event of a module reset (**81.19** = On), Pr **91.05** will default to 3000 rpm.

Plus	Lite V2	Register
✓	✓	✓

Pr 91.06	Speed feedback		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	250 µs

This parameter returns the value of the drive speed feedback in units of 0.01 rpm 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 **03.026** on the Unidrive M is set to a 0 to select the P1 interface or 6 to select the P2 interface on the drive. However, if a low resolution encoder is used there may be some jitter at low speed. For example, at 10 rpm with a 1024 ppr encoder this parameter may jump between 0 and 14.65 rpm. This is similar to the Unidrive M parameter Pr **03.002**.

Plus	Lite V2	Register
✓	✓	✓

Pr 91.07	Current feedback (Pr 04.002)		
Access	RO	Range	Signed 16-bit
Default	N/A	Update Rate	250 µs

The value of Pr **91.07** must be multiplied by Kc / Pr **11.032**. For example when using a Unidrive M700-03400100, Pr **91.07** must be multiplied by 2.22 (i.e.Kc / Pr **11.32**).

Plus	Lite V2	Register
✓	✓	✓

Pr 91.08	Drive analog input 1 value		
Access	RO	Range	±4000
Default	N/A	Update Rate	250 µs

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.

Plus	Lite V2	Register
✓	✓	✓

Pr 91.09	Drive analog Input 2 value		
Access	RO	Range	±1023
Default	N/A	Update Rate	250 µs

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.

Plus	Lite V2	Register
✓	✓	✓

Pr 91.10	Drive analog input 3 value		
Access	RO	**Range**	±1023
Default	N/A	**Update Rate**	250 µs

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.

Plus	Lite V2	Register
✓	✓	✓

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. (See also Note below Pr **91.12**).

Plus	Lite V2	Register
✓	✓	✓

Pr 91.12	Drive analog output 2		
Access	RW	**Range**	±1023
Default	N/A	**Update Rate**	NA

This parameter sets the value of analog output 2. Refer to Pr **91.01** for information on enabling this parameter.

Plus	Lite V2	Register
✓	✓	✓

NOTE When the appropriate shortcut is set (Pr **91.01**), the Second Processor can control the drive's analog outputs directly. The values are not scaled by the drive ±1023 corresponds to ±10 V.

Pr 91.16	Drive digital inputs		
Access	RO	Range	Unsigned 8-bit
Default	N/A	Update Rate	250 µs

This parameter is similar to drive parameter Pr **08.020** 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
5	F6
6	Enable
7	Reserved - Read as zero

Plus	Lite V2	Register
✓	✓	✓

Pr 91.17	Number of Valid CTSync Messages Received		
Access	RW	Range	Signed 32-bit
Default	N/A	Update Rate	

This parameter will increment every time a good CTSync message is received with a valid checsksum.

Plus	Lite V2	Register
✓	×	✓

Pr 91.18	Number of Bad CTSync Messages Received		
Access	RW	Range	Signed 32-bit
Default	N/A	Update Rate	

This parameter will increment every time a CTSync message is received with a bad checksum.

Plus	Lite V2	Register
✓	×	✓

Pr 91.19	Number of Missing CTSync Messages		
Access	RW	Range	Signed 32-bit
Default	N/A	Update Rate	

This parameter will increment every time a message has not been received when the module is expecting one.

Plus	Lite V2	Register
✓	×	✓

Pr 91.20	CTSync Synchronization Signal Width too Short		
Access	RW	**Range**	Signed 32-bit
Default	N/A	**Update Rate**	synchronization

This parameter will increment every time the synchronization signal is the incorrect width. During synchronization this parameter is likely to increase, but should stabilise once synchronization is complete. After synchronization, if this parameter is incrementing then there is likely to be excessive noise on the EIA-RS485 network. In which case it is advisable to check the connections.

Plus	Lite V2	Register
✓	×	✓

Pr 91.21	Inter-option Synchronization Control		
Access	RW	**Range**	0 to 2
Default	0	**Update Rate**	Immediate

This parameter allows the user to set up the Second Processor in the Inter-System Integration Module Synchronization scheme. For more information refer to Chapter 11 *Inter-option synchronization* on page 154.

Bit	Description
0	Set this bit for the Second Processor to participate in the Inter-System Integration Module synchronization scheme as a Producer. Refer to section *NOT CURRENTLY SUPPORTED* on page 154 for details of the term Producer.
1	Set this bit for the Second Processor to participate in the Inter-System Integration Module synchronization scheme as a Consumer. Refer to section *NOT CURRENTLY SUPPORTED* on page 154 for details of the term Consumer.

Plus	Lite V2	Register
✓	✓	✓

Pr 91.22	Inter-option synchronization Status		
Access	RO	Range	Unsigned 8-bit
Default	N/A	Update Rate	Immediate

This parameter shows the status of the Second Processor in the Inter-System Integration Module synchronization scheme. For more information refer to section 11 *Inter-option synchronization* on page 154.

Bit	Meaning	Description
0	Requested Inter-Module synchronization Role	This is identical to bit 1 of the Inter-System Integration Module Synchronization Control Parameter as described above.
1	Requested Inter-Module Synchronization Role	This is identical to bit 0 of the Inter-System Integration Module Synchronization Control Parameter as described above.
2	Inter-Module Synchronization Role Achieved	This bit indicates that for a module attempting to become inter-module synchronization Producer it has achieved that role. In the event of more than one module on a given drive attempting to become the synchronization Producer at least one of these modules will have this bit clear. For a module attempting to become inter-module synchronization 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 synchronization data. (If a Producer has been located in another slot but the rate of the synchronization 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 synchronization data).
3	Synchronization Producer Output within Specification	This bit is relevant only for modules which have been designated as the inter-module synchronization Producer and CTSync Slave. This bit indicates that the signal being provided by the module synchronization 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*

Plus	Lite V2	Register
✓	✓	✓

Pr 91.23	Enable CTSync Output Channels		
Access	RO	Range	Unsigned 8-bit
Default	N/A	Update Rate	Immediate

Enable CTSync output channels on this module with motion engine even if the module is not in CTSync mode. (Note that one of POS0, APC, POS1 must be running for CTSync output channels to be enabled). This parameter has a default value of 1. Setting this parameter to zero can help free up resources if the CTSync output channels are not required.

Plus	Lite V2	Register
✓	×	✓

5.11 Menu 97 - Internal Motion Processor Parameters

Pr **97.00** to Pr **97.99** are 32 bit parameters and are reserved for the internal motion processor. They can be read and written to from the user program, but are not accessible by the keypad.

Plus	Lite V2	Register
✓	✓	✓

5.12 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 Second Processor (and via comms), but may be read-only on the drive's keypad.

Pr 1x.001	Integer, read-write saved on power-down		
Access	RW	Range	Signed 16-bit
Default	0	Update Rate	N/A

This parameter is automatically saved by the drive on power-down.

Plus	Lite V2	Register
✓	✓	✓

Pr 1x.002- Pr 1x.010	Integer, read-only		
Access	RW (RO drive)	Range	Signed 16-bit
Default	0	Update Rate	N/A

The Second Processor has read/write access to Pr 1x.002 to Pr 1x.010, but these parameters are read-only when viewed from the drive's keypad. These parameters are not scalable in the drive's non-volatile memory.

Plus	Lite V2	Register
✓	✓	✓

Pr 1x.011- Pr 1x.030	Integer, read-write		
Access	RW	Range	Signed 16-bit
Default	0	Update Rate	N/A

Plus	Lite V2	Register
✓	✓	✓

Pr 1x.031- Pr 1x.050	Bit, read-write		
Access	RW	Range	0/1
Default	0	Update Rate	N/A

Pr **1x.011** to Pr **1x.050** are saveable in the drive's non-volatile memory.

Plus	Lite V2	Register
✓	✓	✓

Pr 1x.051 – Pr 1x.054	Long integer, read-write saved on power-down		
Access	RW	Range	Signed 32-bit
Default	0	Update Rate	N/A

Plus	Lite V2	Register
✓	✓	✓

5.13 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 Second Processor flash memory upon request. If more than one Second Processor is installed, only one should be configured to store and restore this menu for obvious reasons.

If Pr **81.21** is set, this menu will be saved and restored by the Second Processor.

Pr 20.001- Pr 20.020	**Integer, read-write**		
Access	RW	**Range**	Signed 16-bit
Default	0	**Update Rate**	N/A

Plus	**Lite V2**	**Register**
✓	✓	✓

Pr 20.021 Pr 20.040	**Long integer, read-write**		
Access	RW	**Range**	Signed 32-bit
Default	0	**Update Rate**	N/A

Plus	**Lite V2**	**Register**
✓	✓	✓

6 Communications

6.1 EIA-RS485 Serial Communications Port

See section 2.1 *Features for different module variants* on page 8 for availability of CTSync on your Second Processor.

The Second Processor may come with a EIA-RS485 serial communications port on-board. Refer to Chapter 3 *Installation* on page 12 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**.

6.1.1 CT-ANSI

The Second Processor supports the ANSIx3.28 protocol.

All drive parameters as well as Second Processor parameters can be accessed via the EIA-RS485 port on the module.

6.1.1.1 Reading a parameter

The following tables show the message constructs for reading a parameter.

Table 6.1 Master request

Character	Description
EOT	End of transmission (Ctrl & D)
A1	Second Processor address: 1st digit
A1	Second Processor address: 1st digit
A2	Second Processor address: 2nd digit
A2	Second Processor address: 2nd digit
M1	Menu number: 1st digit
M2	Menu number: 2nd digit
P1	Parameter number: 1st digit
P2	Parameter number: 2nd digit
ENQ	Enquiry (Ctrl & E)

Table 6.2 Slave response (if request was correct and parameter exists)

Character	Description
STX	Start of text (Ctrl & B)
M1	Menu number: 1st digit
M2	Menu number: 2nd digit
P1	Parameter number: 1st digit
P2	Parameter number: 2nd digit
D1 to Dn	Data: 1st digit Data: nth 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. Checksum = M1 ^ M2 ^ P1 ^ P2 ^ D1 ^ D2 ^ ..... Dn ^ ETX. The checksum is an unsigned 8 bit value and if this value is less than 32 then 32 is added to it.

6.1.1.2 Writing to a parameter

The following tables show the message constructs for writing to a parameter.

Table 6.3 Master request

Character	Description
EOT	End of transmission (Ctrl & D)
A1	Second Processor address: 1st digit
A1	Second Processor address: 1st digit
A2	Second Processor address: 2nd digit
A2	Second Processor address: 2nd digit
STX	Start of text (Ctrl & B)
M1	Menu number: 1st digit
M2	Menu number: 2nd digit
P1	Parameter number: 1st digit
P2	Parameter number: 2nd digit
D1 to Dn	Data: 1st digit 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. Checksum = M1 ^ M2 ^ P1 ^ P2 ^ D1 ^ D2 ^ ..... Dn ^ ETX. The checksum is an unsigned 8bit value and if this value is less than 32 then 32 is added to it.

6.1.1.3 Second Processor address

The Second Processor will only act on messages received where the full address matches the address of the Second Processor or the group address in the message (1^{st} digit) matches the 1^{st} digit of the address of Second Processor or the address in the message is a broadcast (0). Broadcast messages are used to write data to multiple nodes.

6.1.1.4 Control Characters

Table 6.4 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

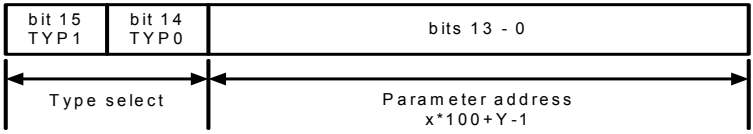
6.1.2 Modbus-RTU

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 **01.021** will be register number 40121. Only parameters up to **MM.99** are accessible on the Unidrive M target via the RS485 port on SI-Applications Plus.
Drive parameters accessed using Modbus and the SI-Applications Plus RS485 port cannot be set using the **MM.PPP** style mappings for Unidrive M, only **MM.PP.** Where Pr **1.021** is to be accessed remove the first zero after the decimal point to give a reference of 121, where 121 could be entered in Pr **81.27**, Pr **81.29**, Pr **81.31** to Pr **81.34**.

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:



Type field bits 15-14	Access
00	16-bit. Backwards compatible
01	32-bit.
10	Reserved
11	Reserved

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 that 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
- RtuPresetMutlipleParas
- RtuMasterReply
- RtuMasterStatus

6.1.2.1 FC03 Read multiple registers

Read a contiguous array of 16 bit 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.s

The following tables show the message constructs for Modbus RTU Function Code 03

Table 6.5 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 16 bit registers MSB
5	Number of 16 bit registers LSB
6	CRC LSB
7	CRC MSB

Table 6.6 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

6.1.2.2 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 32 bit parameter but only 16bits of data can be sent.

The following tables show the message constructs for Modbus RTU Function Code 06.

Table 6.7 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 6.8 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

6.1.2.3 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 6.9 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 6.10 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

6.1.2.4 FC23 Read/Preset 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 6-5 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
12	Register data 0 LSB
11 + byte count	CRC LSB
12 + byte count	CRC MSB

Table 6.11 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

6.1.3 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. Note that the 32-bit access mode is specific to Control Techniques.

6.1.4 User Modes

These modes turn off all internal protocols and allow the user to access the EIA-RS485 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.

6.2 CTNet

See section 2.1 *Features for different module variants* on page 8 for availability on your module. Full details of CTNet are outside the scope of this User Guide and can be found in the separate *CTNet User Guide*.

6.3 Second Processor Mapping Parameters (fieldbus)

NOT CURRENTLY SUPPORTED

The Second Processor 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 6.12 Second Processor Internal Parameters

Second Processor Parameters	Parameter Reference	DIrect to Slot 1	DIrect to Slot 2	DIrect to Slot 3
_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, Plus & ST Plus Only)	Pr **85.XX**	Pr **115.XX**	Pr **145.XX**	Pr **175.XX**
DIgital I/O Parameters (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 Second Processor. The fieldbus interface module can read data from and write data to an Second Processor installed in any slot in the drive, simply by specifying the target parameter as shown in Table 6.12.

NOTE With a single Second Processor is installed to the drive, normal Second Processor parameter references can be used without problem, as the fieldbus will automatically divert them to the Second Processor.

6.3.1 Example configuration 1

Consider a drive with the following configuration:

- Slot 1 - Vacant
- Slot 2 - Second Processor
- Slot 3 - PROFIBUS module

If a parameter read request comes over the PROFIBUS network to read Pr **71.08**, this will be re-directed to the Second 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 network to read Pr **131.08**, this will be sent straight to the Second Processor in slot 2. The value in _Q08% from slot 2 will be returned.

If a parameter read request comes over the PROFIBUS network to read Pr **101.08**, this will be sent straight to the Second Processor in slot 1. As there is no Second Processor installed in slot 1, an error message will be returned, indicating that the parameter does not exist.

If for example a 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.

7 DPL Programming

This chapter covers:

- Basic DPL program structure and syntax
- Basic DPL commands
- New features offered by the Second Processor

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 #1.021
$DEFINE PRESET_REF_2 #1.022
$DEFINE SPEED_FB #3.002
```

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.

NOTE To give backward compatibility with the Unidrive SP **MM.PP** style parameter access, the SI-Applications supports both **MM.PP** and **MM.PPP** style parameter access e.g. 01.21, 1.21, 01.021, 1.021 are all valid ways to access Unidrive M Pr **1.021**.

7.2 Tasks

A DPL program is separated into separate 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 the 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 1-200 ms) 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 µs to 8 ms). 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 System Integration 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).

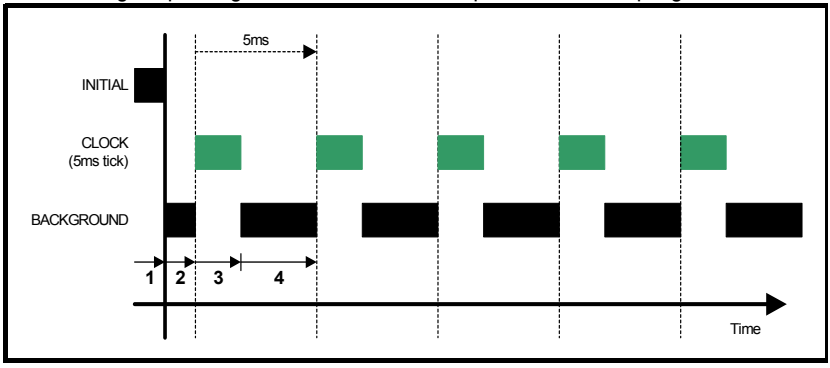
NOTE The UD70 ENCODER and SPEED tasks can still be used. These are now aliases for the POS0 and POS1 tasks respectively (i.e. if the program has an ENCODER task this is the same as if it contained a POS0 task) . The timebase for both tasks are not fixed as in the UD70 but specified by the user. The CLOCK task in the Second Processor 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:

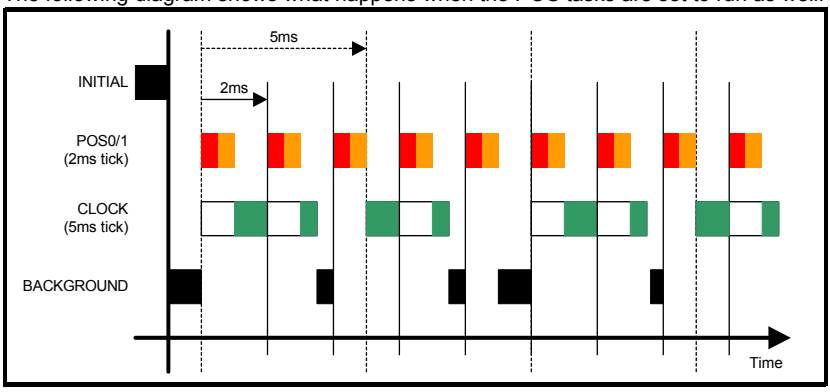


Key:

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 Second Processor is.

NOTE The Plus module and V2 module provide Pr **88.03** through to Pr **88.08** which will give a greater accuracy on the available resources.

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.	±3.40282e+038
Double float	64-bits: 1 sign bit, 52-bit mantissa, 11 bit exponent	±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 Initialization 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 Second Processor. Remember that resetting the drive from a tripped condition will also cause a reset (depending on the value in Pr **81.15**).

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 384 kb - 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 "Result% = 1"
ENDIF
```

Example 2 (arrays):

```
DIM MyArray%[10]
...
IF MyArray%.1[4] = 1 THEN;test bit 1 of element Pr 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 Second Processor *Flash* memory.

See section 5.4 *Menus 70-75 - PLC Registers* on page 40 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 Second Processor. 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 categorized into two sets:

- Drive Parameters
- Second Processor 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 Second Processor parameters are local and accessible only to the Second Processor. These parameters provide access to additional features of the Second Processor, and give faster access to some drive parameters.

NOTE The Second Processor always guarantees that the drive parameter database it uses matches that of the host drive. When a Second Processor is installed to a Unidrive M for the first time and powered up the word "Waiting For Options" may appear on the drive display for a few seconds. This indicates the Second Processor is synchronizing databases. This will only occur the first time the module is installed to the drive.

7.4.1 Reading and Writing Parameters

Reading and writing parameters is achieved by using the **#** command. Parameters may be accessed in Pr **MM.PP** format as they were on Unidrive SP where the leading zero may be removed from the parameter, or in Pr **MM.PPP** format as they are on the Unidrive M keypad.

For example, to read the speed feedback parameter (Pr **03.002**), use:

```
Speed% = Pr 3.002
```

To write to a speed reference parameter (e.g. Pr **01.022**), use:

```
Pr 01.022 = 1500
```

Note that the leading zeros in the menu/parameter field are optional. For example Pr **3.002**, Pr **03.002**, Pr **3.002**, Pr **03.02**, Pr **3.02**, Pr **03.2** and Pr **3.2** will access exactly the same parameter.

This gives backward compatibility of software written for Unidrive SP, since most Unidrive M parameters directly overlay with those of Unidrive SP.

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 **01.019** has a range of 0.000 - 0.099. Reading the parameter using:

```
Speed_Fine% = #INT1.019
```

will return integer values between 0 and 99. When writing, the command:

```
#INT1.019 = 45
```

will set the parameter to 0.045 (same as Pr **01.019** = 0.045). The benefit of this is that the DPL program can use integer variables (%) instead of floating-point, thus providing a speed advantage.

NOTE The #INT directive can be used with **MM.PP** parameter format as they were on Unidrive SP where the leading zero may be removed from the parameter, or in **MM.PPP** format as they are on the Unidrive M keypad.

7.5 Operators

DPL offers all the standard operators as follows:

Table 7.3 Standard operators in order of precedence

Operator	Meaning
-	Arithmetic negation
!	Boolean negation (unary)
!(..., nbit)	Negation of *nbit* bits
*	Multiplication
/	Division
%	Modulo (remainder)
+	Addition
-	Subtraction
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR (XOR)

Table 7.4 Conditional operators in order of precedence

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

The DPL language implemented for the Second Processor is backwardly compatible with the UD70 product for Unidrive Classic, but does have a few new extensions.

Refer to the on-line help for full reference on the DPL language and function block library.

7.6.1 New Commands for Second Processors

These are new commands that have been introduced into the DPL language for the Second Processor and were not present in the UD70 for Unidrive Classic.

FOR loop

This is new to the Second Processor.

```
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 recognized 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
```

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 Pr 90.13 = 45 THEN
    // task scheduled from DPL
ENDIF
}
```

CTNETDIAGNOSTICS

This is only available on Second Processors with CTNet. Please see the Features section on pages 8 & 9 to see if CTNet is available on your Second Processor.

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 overleaf details the outputs and the equivalent UD70 variable.

CTNetDiagnostics Output Values

Output	Value
1	Total number of messages handled by this node
2	Number of cyclic data overruns
3	Lost RX messages
4	Number of retries
5	Number of recons
6	Number of excessive NAKs
7	Duplicate sync
8	Number of locally generated recons
9	Number of non-cyclic messages
10	Number of lost routing messages

GETPARATTR

This is used to get parameter attributes such as maximum and minimum values, read-only flag, etc.

```
(Max%, Min%, Flags%) = GETPARATTR(Menu%, Par%)
```

NOTE The Par% input to the GETPARATTR function block accepts both **MM.PPP** Unidrive M and **MM.PP** Unidrive SP parameter references e.g. Par% = 001 and Par% = 01 would both access MM.001 on Unidrive M.

CModeXfer

This allows the user to change the drive mode without any installed System Integration 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 and 24 to 28 will not be defaulted when using this command.

RtuReadRegs

NOTE Only MM.PP parameter access is supported for this RS485 communications protocol, therefore to read from or write to a Unidrive M parameter it is necessary to omit the leading zero from the parameter identifier e.g. Pr **1.021** may be identified as 121 or Menu% = 1 Par% 21.

RtuReadParas
RtuReadInputRegs
RtuPresetRegs
RtuPresetParas
RtuMasterReply
RtuMasterStatus

These commands are implemented to allow the user to utilize the SI-Applications Plus and SI-Register modules 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 Pr 81.49 to 101.
```

ANSIREPLY

This command existed in the UD70, however the syntax has changed for the Second Processor.

```
(Status%, Reply%) = ANSIREPLY()
```

It is a EIA-RS485 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 initialize 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% = Pr 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

NOTE Only **MM.PP** parameter access is supported for this RS485 communications protocol, therefore to read from or write to a Unidrive M parameter it is necessary to omit the leading zero from the parameter identifier e.g. Pr **1.021** may be identified as 121 or Menu% = 1 Par% 21.

AssRAM
UnassRAM
RamLength
SetRamLength

These commands allow the programmer to use the RAM files within the Second Processor. 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.

7.6.2 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 UDFB's. 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 UDFB's 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, UDFB's 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 UDFB's.

A UDFB can of course access drive parameters and Second Processor 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%) {
// Initialization code:

Scale% = 500     // initialise a local variable

FBbody
// main body code:

Output% = Input1% + Input2% * Scale% / 1000

}
```

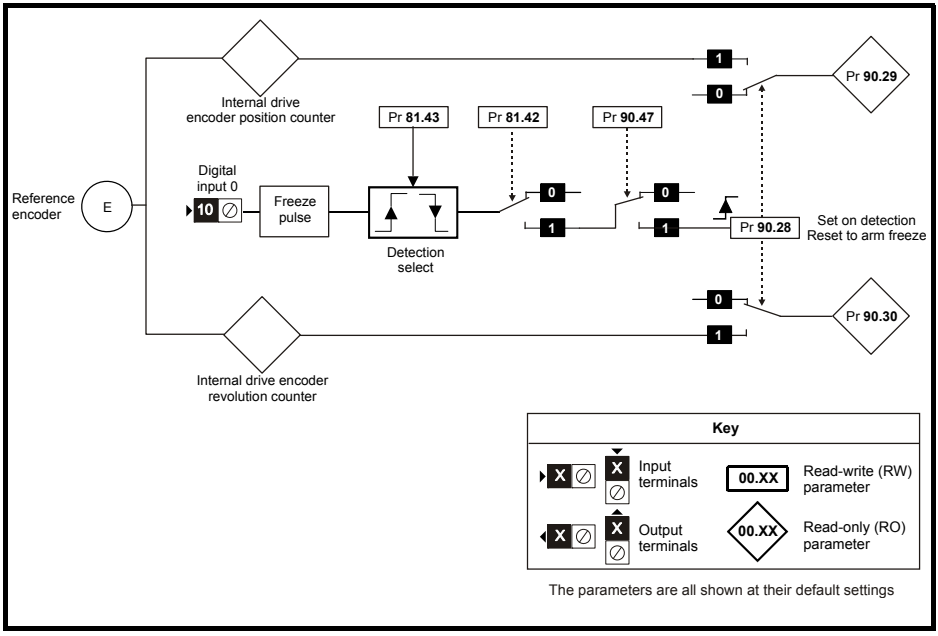
8 Freeze and marker

8.1 Freeze input (SI-Applications Plus and SI-Register)

See section 2.1 *Features for different module variants* on page 8 for availability of CTSync on your Second Processor.

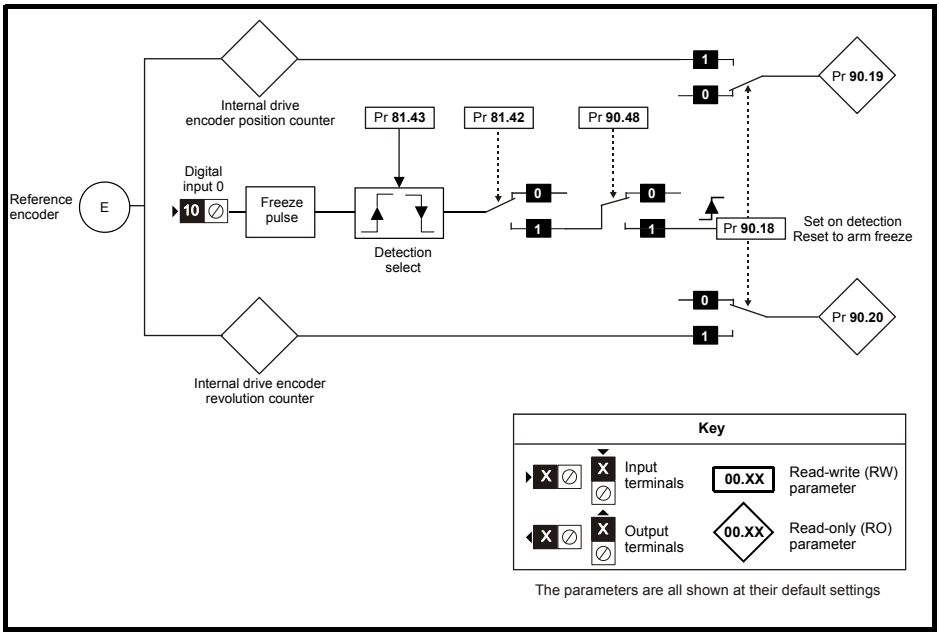
Digital Input 0 (DIGIN0) can be used to 'freeze' the reference and feedback encoder counters.

Figure 8-1 SI-Applications Plus and SI-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 8-2 SI-Applications Plus and SI-Register Feedback Freeze Input



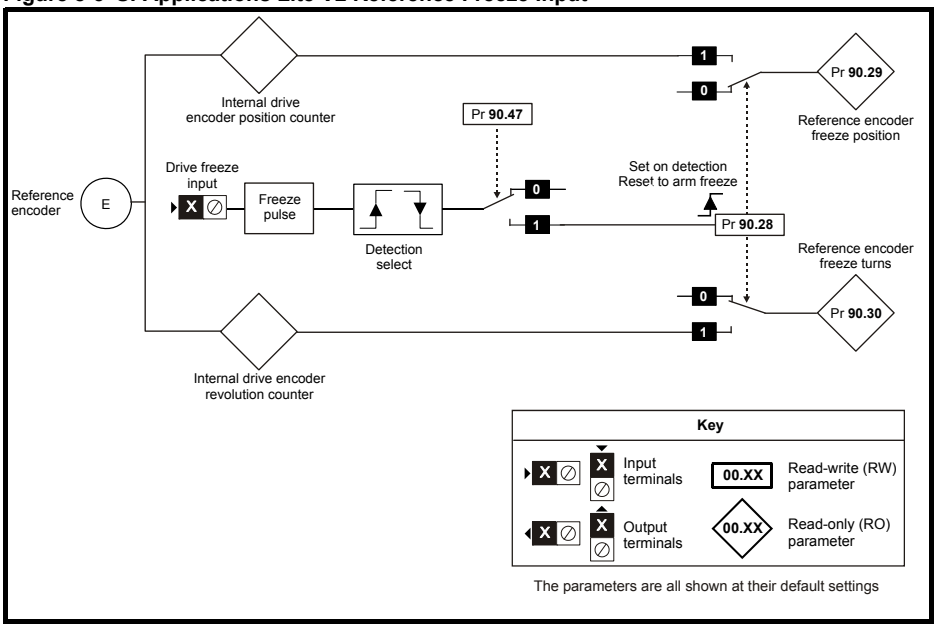
The encoder revolution counter is cached into parameter Pr **90.20** and the encoder position is cached into Pr **90.19**.

NOTE When Pr **81.42** is set to 1, Pr **03.100** on the drive is set to 4. This sets the common freeze line as the F1 freeze trigger source in the drive.

8.2 Freeze input (SI-Applications Lite V2)

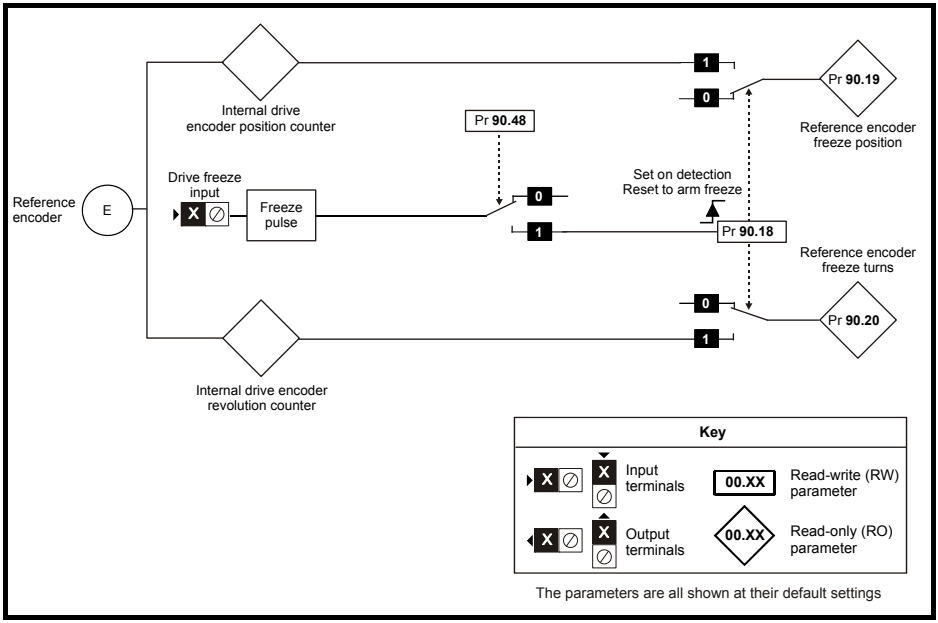
The SI-Applications Lite V2 is not like the SI-Applications Plus and SI-Register modules in that it does not have digital I/O which can be used to 'freeze' the reference and feedback encoder counters. However, another module in the same drive or the Unidrive M drive terminals can be used to freeze the encoder data in the modules. For instance, setting parameter **3.100** for the Unidrive M to 0 allows Digital IO 4 to be used as an input to freeze data

Figure 8-3 SI-Applications Lite V2 Reference Freeze Input



The encoder revolution counter is cached into parameter Pr **90.29** and the encoder position is cached into Pr **90.30**.

Figure 8-4 SI-Applications Lite V2 Feedback Freeze Input



The encoder revolution counter is cached into Pr **90.19** and the encoder position is cached into Pr **90.20**.

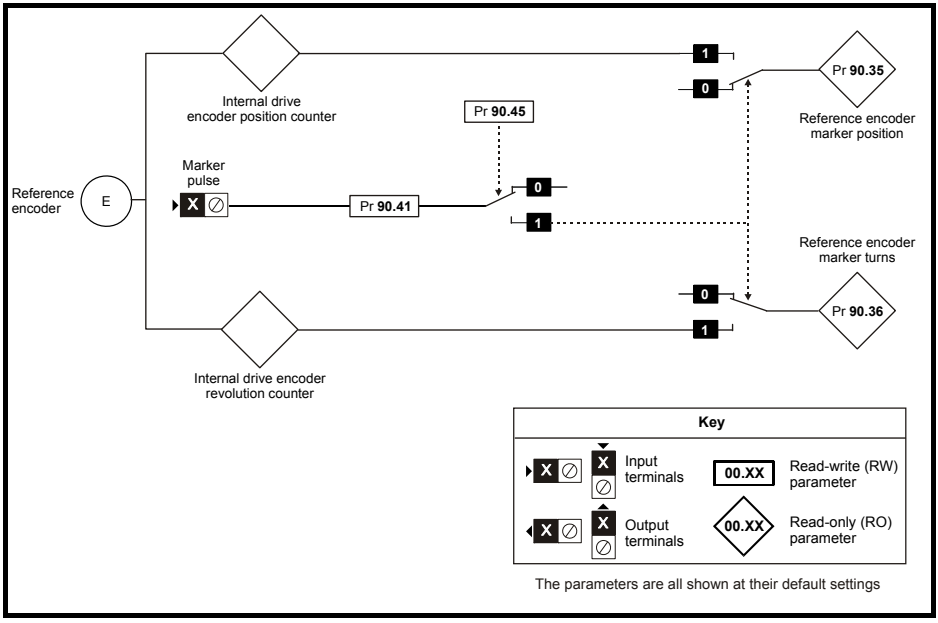
The freeze position of both the reference and feedback encoders can be captured on the rising or the falling edge of the freeze pulse. This is determined by setting parameter Pr **81.43** to either 0 (rising edge) or 1 (falling edge). Parameter Pr **81.42** enables the position to be written to Pr **90.19** and Pr **90.29** and the revolution counters to be written to Pr **90.20** and Pr **90.30**.

When a freeze input is seen Pr **90.18** and Pr **90.28** are set to 1 automatically so that the position can be written to Pr **90.19** and Pr **90.29** and the revolution counters can be written to Pr **90.20** and Pr **90.30**. Pr **90.18** and Pr **90.28** must be reset to zero if the user wants to update the data again on the next freeze pulse.

8.3 Marker pulse

The Second Processor is able to cache the position and revolution count at the point when a Marker pulse is seen on the reference or feedback encoders.

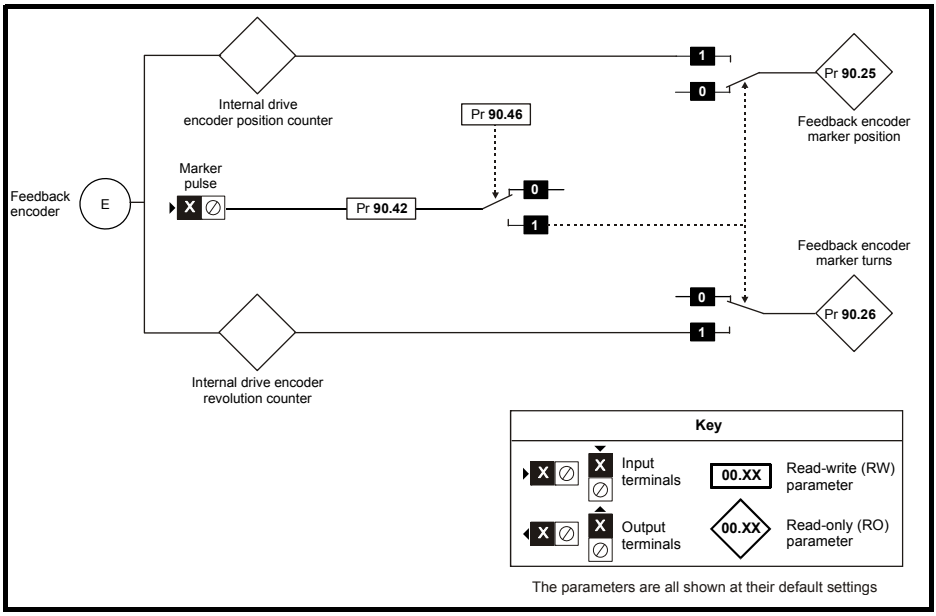
Figure 8-5 Second Processor Reference Marker



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 8-6 Second Processor 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.

9 CTSync

9.1 Overview

See section 2.1 *Features for different module variants* on page 8 for availability of CTSync on your Second Processor.

The Second Processor may be used to synchronize 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 32-bit integers and 1 x Unsigned 8-bit integer.

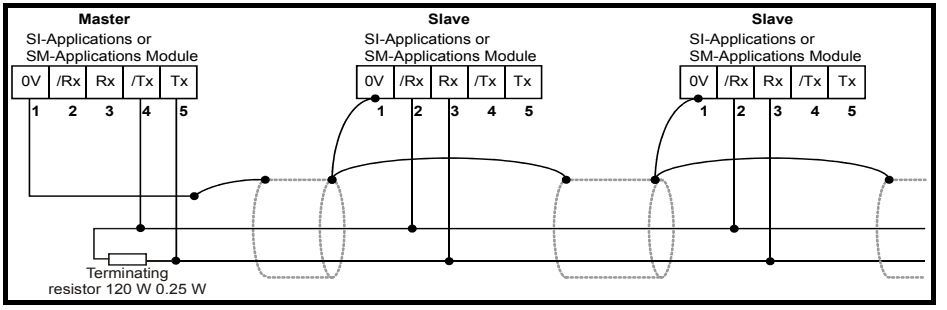
Only one Second Processor should be configured as the Master and all others configured as Slaves if they need to participate in the CTSync scheme. The Master generates reference data which is transmitted to all Slaves on the network. The Master can be set to operate as a Slave, if for instance two drives 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.

9.2 Connections

CTSync operates via a connection between the EIA-RS485 ports of the Second Processors on the network in either 2-wire or 4-wire. Refer to section 3.6 *EIA-RS485 Connections* on page 16 for information on how to connect the SI-Applications Modules RS-485 ports.

To simplify wiring the Slave transmit and Master receive signal line connections can be omitted in 4-wire mode (See Figure 9-1). This is because the Master does not receive a response from the Slave.

Figure 9-1 CTSync Wiring Example for SI-Applications



9.3 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-RS485 network is 1200 m.

9.4 CTSync Function Blocks

Six Function Blocks are available when using the CTSync functionality of the Second Processor. These are listed below.

9.4.1 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.

9.4.2 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.

9.4.3 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.

Parameter%: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

NOTE Par% directive can be used with **MM.PP** parameter format as they were on Unidrive SP where the leading zero may be removed from the parameter, or in **MM.PPP** format as they are on the Unidrive M keypad.

9.4.4 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.

9.4.5 CTSYNCDisableOutputChannel

```
Result% = CTSYNCDisableOutputChannel(Channel%)
```

This function block causes the specified channel to cease writing it’s data to the Unidrive M.

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.

9.4.6 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.

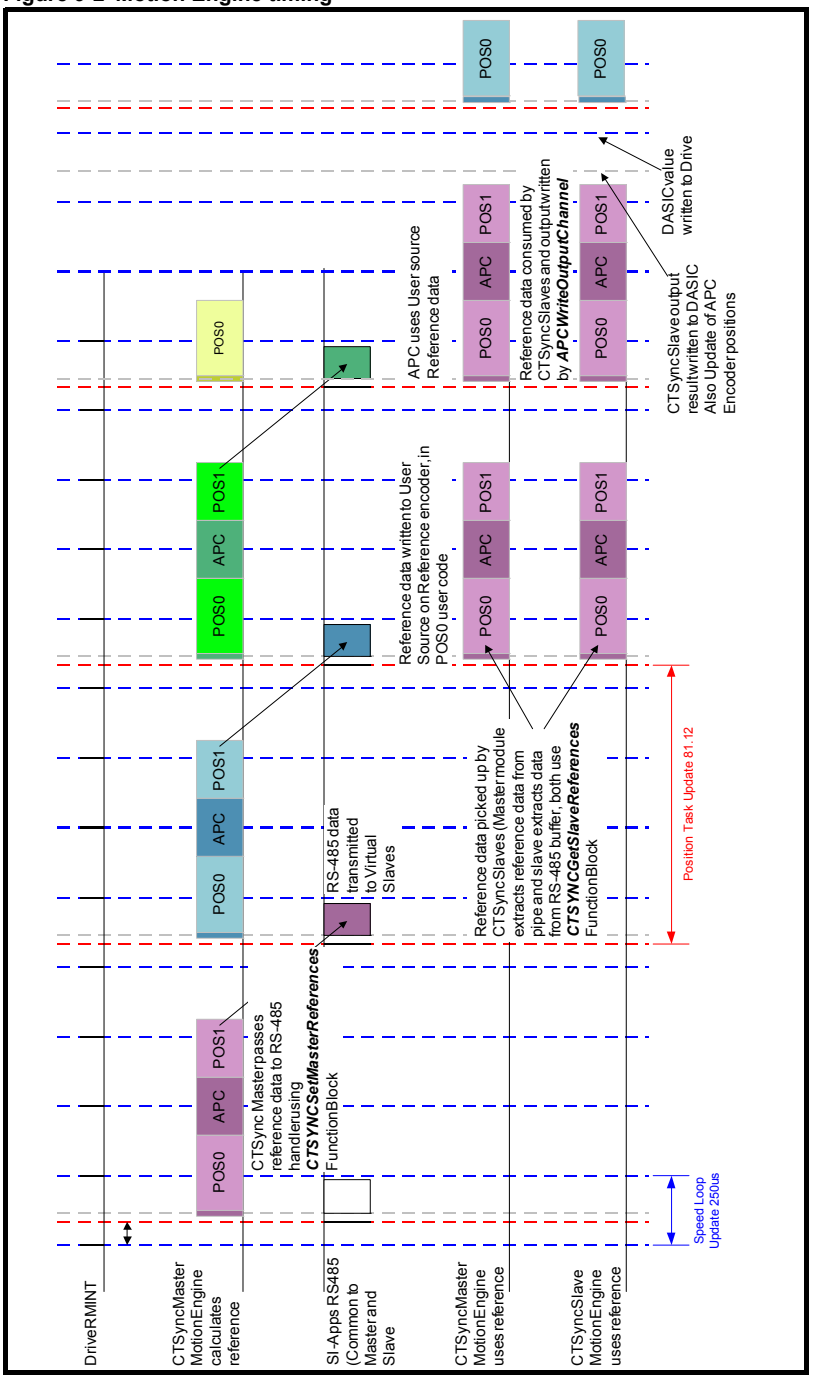
9.5 Motion Engine

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 9-2 *Motion Engine timing*

Figure 9-2 Motion Engine timing



9.6 Virtual Master Example

9.6.1 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)

}
```

9.6.2 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
}
```

10 SI-Register functionality

10.1 Introduction to SI-Register

The SI-Register module offers all of the functionality of an SI-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.

10.1.1 Application uses for the SI-Register module

The flexibility of the SI-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 SI-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.

10.1.2 Features

The SI-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.

10.1.3 Advantages over standard freeze

The Freeze scheme present on the SI-Applications range of modules, and the SI-Register module, has similar but far more limited capturing and data manipulation capabilities to those present in the SI-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 SI-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 10.1 Burst edge capture for SI-Register

Burst edge capture for SI-Register (single channel active)	
Position Update Rate	Maximum Number Of Edges
250 µs	8
500 µs	10
1 ms	25
2 ms	50
4 ms	100
8 ms	200

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

Table 10.2 Sustained edge capture for SI-Register

Sustained edge capture for SI-Register (single channel active)	
Position Update Rate	Microseconds Per Edge
250 µs	80
500 µs	70
1 ms	60
2 ms	60
4 ms	60
8 ms	60

NOTE The data above represents the peak expected values while using the SI-Register system; the actual throughput in a real application may be lower where large volumes of additional code are also being handled by the SI-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 10.1 and Table 10.2 it can be seen that for a nominal position update rate of 1ms, the SI-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 SI-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 SI-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 SI-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 SI-Register system and only need to be configured to suit the application.

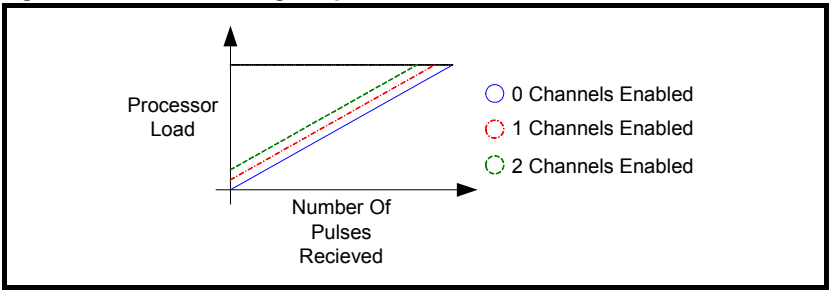
NOTE The SI-Register and Freeze capturing systems are entirely separate entities, and can both be used at the same time on SI-Register.

10.1.4 Considerations when using SI-Register

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

- Processing resource used is proportional to the number of pulses the SI-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 SI-Register module will completely run out of processing resource, causing a Slotx Error (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.

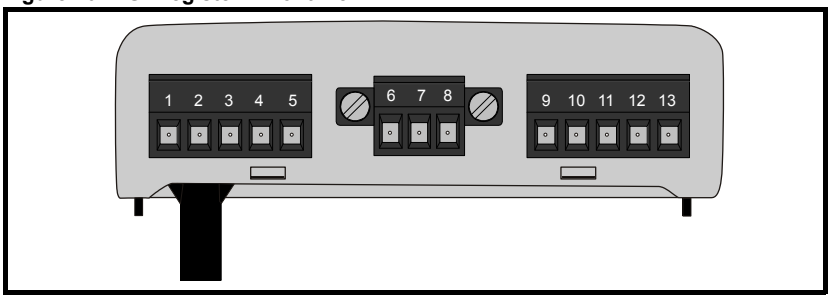
Figure 10-1 Processor usage vs pulses received



- The SI-Register module is capable of everything that an SI-Applications Plus module can do, however it is recommended that the SI-Register module is only used to perform the registration position capture and offset calculation, and that an additional SI-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 SI-Register module may be able to handle the complete application; the user is responsible for testing their system to verify this.
- The SI-Register capture functionality and the Menu 85 timers function are mutually exclusive, therefore when the SI-Register is scheme is enabled the timer will not be available to the user.
- The method of determining the captured event positions within the SI-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 4096 ppr incremental encoder accelerating from 0 to 2000 rpm at a rate of 10000 rpm/s.

10.2 Registration sensor connections

Figure 10-2 SI-Register - Front view



The terminal functions are given in Table 10.3.

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

Table 10.3 Module connectors

Terminal	Function	Description
9	0 V	0 V connection for sensor inputs.
10	DI0	SI-Register Channel 0 sensor input.
11	DI1	SI-Register Channel 1 sensor input.

Table 10.4 Digital input specifications

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

10.3 Functional description

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 <250 µs (ideally in the order of 1 µs), to allow the SI-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 1 µs, however the falling edge time can be much longer depending on the pull down resistor used. This is particularly important in high speed systems.

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

10.3.1 Sensor setup

In order to get the best performance from the SI-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 40 µs, observes a single mark on a drum of circumference 500000 µm. When the drum turns at 1m/min the position captured with respect to 1 revolution is 200000 µm. At 600 m/min however the response time in the sensor causes an offset equal to (600 m/min x 1000000 x 40 µs) / (60 x1000000) = 400 µm or 0.4 mm, giving a captured position of 200400 µm.

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 stabilizes, make a note of the position values.
5. Change the axis speed from 10 rpm to the maximum speed possible e.g. 3000 rpm.
6. Follow step 4, making a note of the stabilized 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 x (1/((3000 rpm/60) * UPR))

   (UPR is Units per revolution).
9. Convert the result to time in nanoseconds, ready to configure the SI-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 139 ns interval; 139 ns 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 <250 µs (ideally in the order of 1 µs), to allow the SI-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 1 µs, 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 16 µs, observes a single mark on a drum of circumference 500000 µm. When the drum turns at 1000 m/min the position output may vary by up to (1000 m/min x 1000000 x 16 µs) / (60 x1000000) = 267 µm or 0.267 mm.

10.3.2 DPL code

When writing the SI-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 SI-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 SI-Register code, since a Slotx Error trip (TR54 - DPL Task Overrun) is much more likely to occur.

When writing software using the SI-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 SI-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).

10.3.3 Choosing a position data source for capture

Each SI-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 SI-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:

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

To select one of the 12 available position data sources set Pr **90.43**, Pr **90.44** or Src% from 0 to 11, where:

Value	Description
0	P1 Drive encoder
1	P1 Slot 1
2	P1 Slot 2
3	P1 Slot 3
4	User program
5	Unconfigured
6	P2 Drive encoder
7	P2 Slot 1
8	P2 Slot 2
9	P2 Slot 3
10	P1 Slot 4
11	P2 Slot 4

If the Reference or Feedback encoder object directed to the user program (Pr **90.44** = 4, or Pr **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.

10.3.4 Choosing position units

Each SI-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 500000 µm 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 $2^{(32 - \text{Turns Bits})}$ 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^{16} 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 = $2^{(32 + \text{PositionUnits\%})}$

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} / 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^{31}-1 in the forwards direction, or past -2^{31}in the reverse direction.

Scaling the position units in this way enables any position errors / offsets calculated by the SI-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.

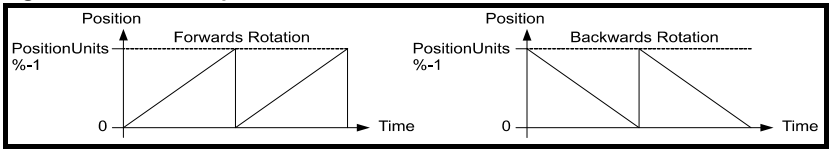
If the unit scaling of the motion platform differs from the SI-Register scaling, then a conversion calculation will be required in the DPL Code before the offset can be used.

10.3.5 Choosing the position format

The SI-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 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).

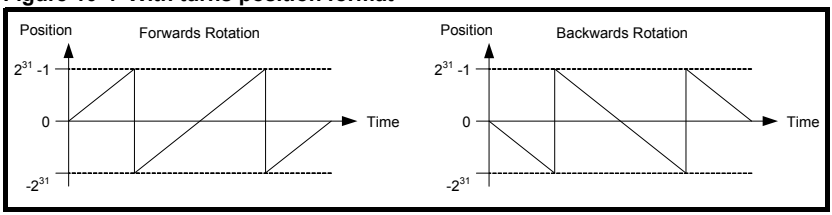
Figure 10-3 No turns position format



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 10-4 With turns position format



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

To configure the SI-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} / 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.

10.3.6 Choosing an operating mode

The SI-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).

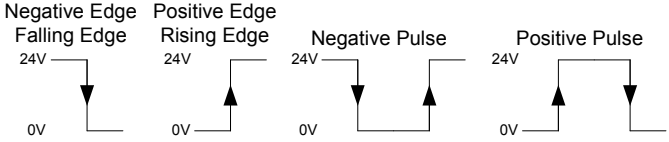
Automated capture: The automated capture types are for pulse pattern recognition or frame capture. In the automated modes, data is collected from the system using the ***GetRegistrationFrame*** function block. The user must define the pattern of pulses or data frame to be captured before enabling capture 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 24 V to 0 V transition.
- A positive edge is defined as a 0 V to 24 V transition.
- A negative pulse is defined as a 24 V to 0 V transition, followed by a 0 V to 24 V transition.
- A positive pulse is defined as a 0 V to 24 V transition, followed by a 24 V to 0 V transition.

Figure 10-5 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 synchronizing print rollers, or sheeting printed products where the print register marks are used to define where the sheet cut point is.

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

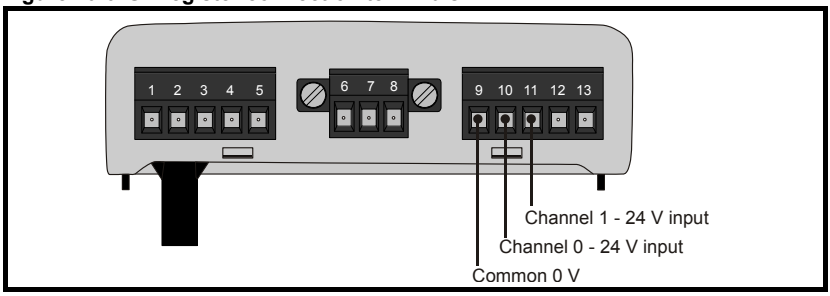
- OperationMode% = 1, manual negative pulses are selected.
- OperationMode% = 2, manual positive pulses are selected.
- OperationMode% = 3, automated negative pulses are selected.
- OperationMode% = 4, automated positive pulses are selected.
- OperationMode% = 5, manual negative edges are selected.
-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

10.3.7 Choosing a capture channel

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

Figure 10-6 SI-Register connection terminals



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

To configure the SI-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.

10.3.8 Setting up automated capture

The SI-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 30 mm from previous pulse and maximum of 100 mm.
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.

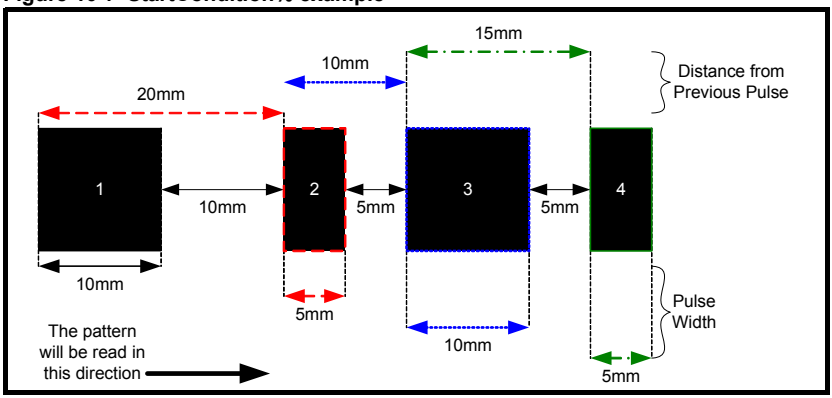
NOTE The minimum and maximum distance from the previous pulse is specified in this way to minimise the effect material stretch, when looking for patterns of pulses and gaps e.g. when printing on to plastic film. Other methods of specification like using the distance from the first pulse to specify the 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 (0.001 mm) units.

Figure 10-7 StartCondition% example



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

To configure the SI-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.

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

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

When the SI-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.

10.3.10 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 SI-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 SI-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 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).

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.

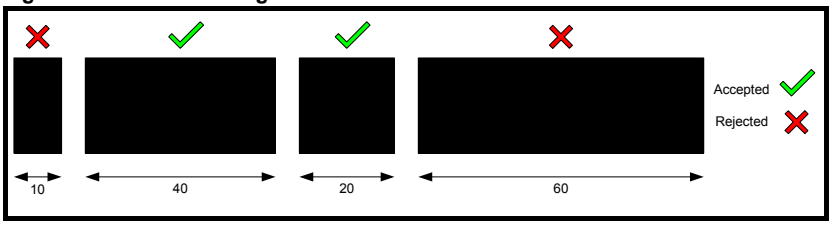
NumberOfEvents% - This displays the number of complete pulse results that will be seen in the OutputArray%. Under normal circumstances NumberOfEvents% = StartCondition%[1] used by the ***SetRegistrationStartCondition*** function block.

10.3.11 Filtering event data

The SI-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.

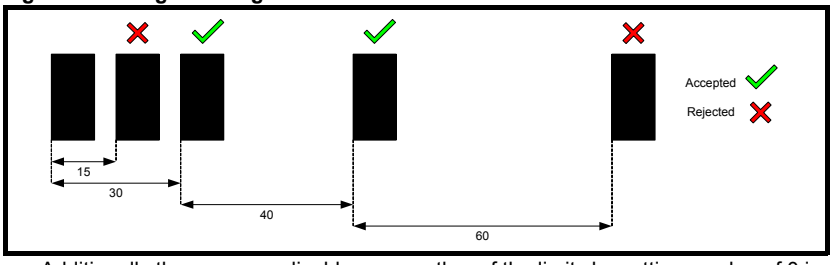
Figure 10-8 Pulse filtering



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 <= MaximumDistance% are reported, or if MaximumDistance% = 0, then pulse widths >= 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 10-9 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 <= MaximumDistance% are reported, or if MaximumDistance% = 0, then pulse widths >= 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.

10.3.12 Windowing event data

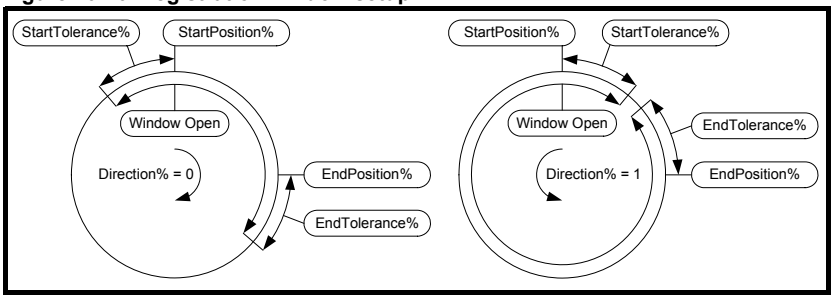
The SI-Register position capture system has a comprehensive windowing scheme that automatically accepts complete events (pulses and edges), that are seen inside a configurable position window (StartPosition% to EndPosition%), and rejects all events outside of the window without the need for extensive 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 SI-Register processor resource will not be completely used, preventing an Slotx Error trip (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 10-10 Registration window setup



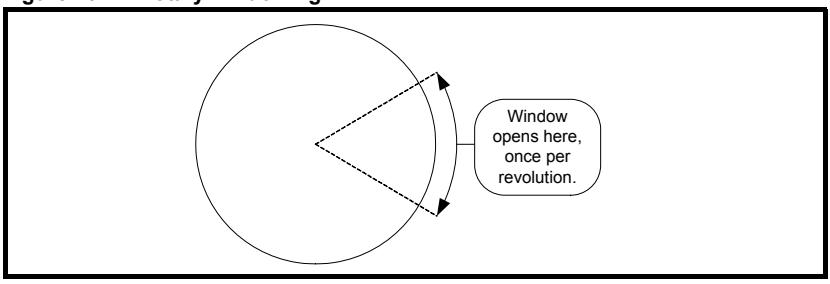
In Figure 10-10 it can be seen that for the forwards direction (Direction% = 0), that the window open area is defined by (EndPosition% + EndTolerance%) - (StartPosition% - StartTolerance), and that for the reverse direction (Direction% = 1), the window open area is defined by (StartPosition% + StartTolerance) - (EndPosition% - 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 10-11 Rotary windowing



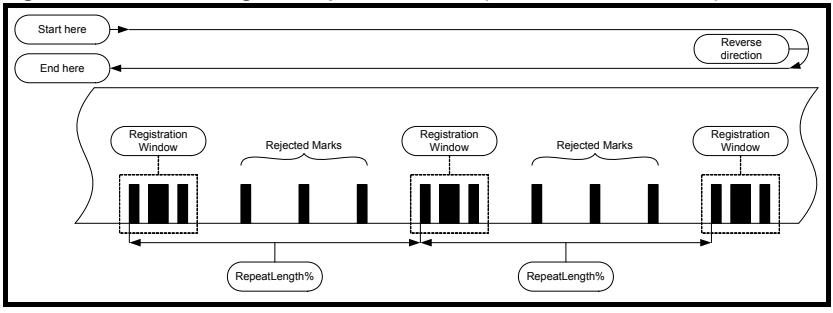
NOTE The window must be long enough to be open for at least 3 position update cycles at any given speed, e.g. an application runs at 600 m/min, and a position update cycle of 1 ms, giving a minimum window length of:

Min Length = (600 / (60 x 1000)) x 3 = 0.03 m = 30 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 10-12 Windowing with repeatdistance% (with turns information)

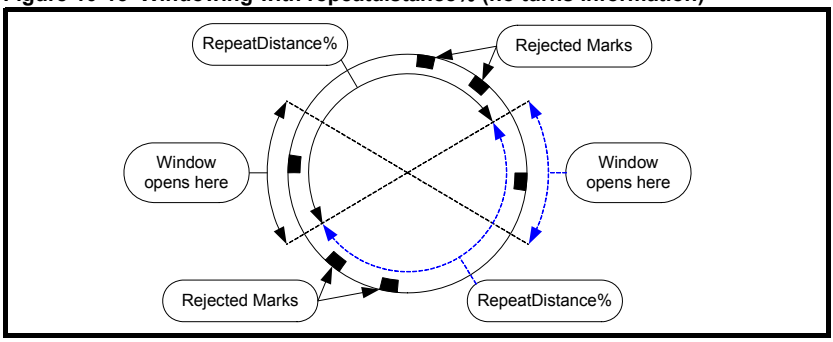


When turns information is selected, RepeatDistance% must be >= total window length to move the window automatically. To prevent the repeat distance from moving the window, set RepeatDistance% to 0.

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^{χ} units per revolution, to the application measurement units e.g. mm.

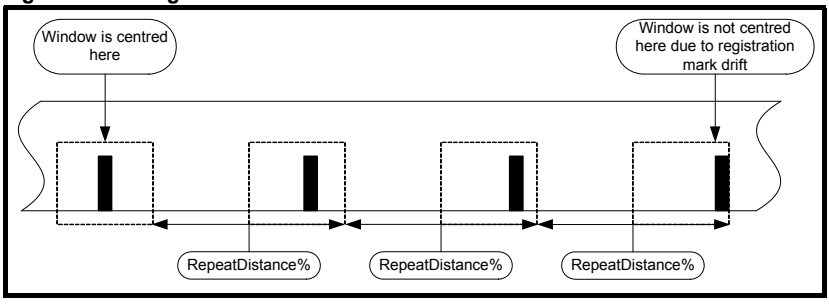
A RepeatLength% may also be set for applications where no turns information is specified. The RepeatDistance% must be >= total window length and <= PositionUnits%. This may be useful in situations where there is more than one event per revolution, and several marks to be disregarded inbetween. See Figure 10-13.

Figure 10-13 Windowing with repeatdistance% (no turns information)



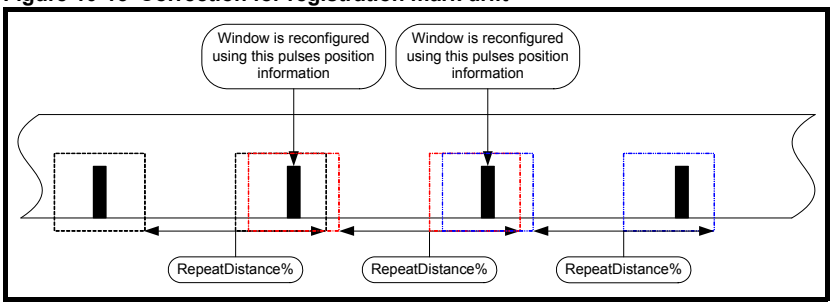
In most linear applications there will be small variations in the distance between windowed events, caused by slight changes in printed position, stretch in the material or even roller slip, that will cause the window to go out of synchronisation with the marks on the product; this is called registration mark drift. See Figure 10-14.

Figure 10-14 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 10-15 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 SI-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.

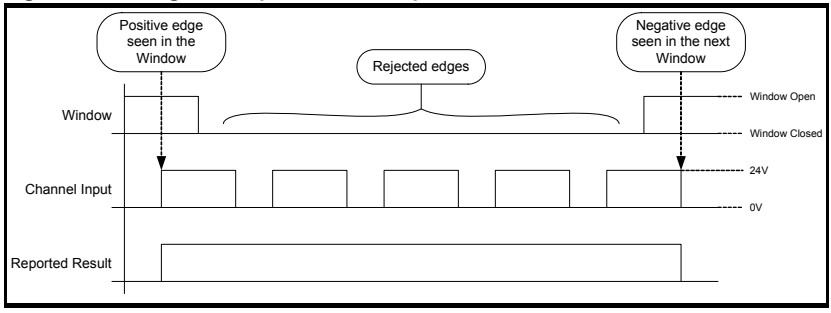
The user may also calculate and implement a window in advance if preferred.

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 the user will be the distance between these two edges, even though the window has closed then opened in between. See Figure 10-16.

Figure 10-16 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.

10.3.13 Monitoring functions blocks for SI-Register

The SI-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 139 ns.

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 SI-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 SI-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.

10.3.14 Position and distance calculation

The SI-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.

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

Where an SI-Register module trips the drive on a Slotx Error 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 SI-Register module to an additional SI-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 SI-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 Pr **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 SI-Register function block calls a higher priority, since the Clock task has a higher priority than the Background task.

NOTE Provided only SI-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.

10.4 Registration function blocks

10.4.1 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 SI-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 Pr **90.44** or APCSetReferenceSource(Src%) for the reference encoder, or Pr **90.43** or APCSetFeedbackSource(Src%) for the feedback encoder. To select one of the 12 available position data sources set Pr **90.43**, or Pr **90.44** or Src% from 0 to 11, where,

Value	Description
0	P1 Drive encoder
1	P1 Slot 1
2	P1 Slot 2
3	P1 Slot 3
4	User program
5	Unconfigured
6	P2 Drive encoder
7	P2 Slot 1
8	P2 Slot 2
9	P2 Slot 3
10	P1 Slot 4
11	P2 Slot 4

If the PositionDataSource% is the user program (Pr **90.44** = 4, or Pr **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 SI-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 SI-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 PositionUnits% > 0: 0 to (PositionUnits% - 1).
- If PositionUnits% <= 0: $-2^{(32 + \text{PositionUnits\%})}$ to $2^{(32 + \text{PositionUnits\%})}-1$.

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} / 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.

`ChannelNumber%`

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

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

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151.

10.4.2 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 SI-Register digital input [0].

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

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151.

10.4.3 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 SI-Register Digin[0].

1: Capture Channel 1, or SI-Register Digin[1].

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151.

10.4.4 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%)
```

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 SI-Register digital input [0].

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

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151.

10.4.5 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 SI-Register digital input [0].

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

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151.

10.4.6 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 SI-Register digital input [0].

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

Return values

`Status%`

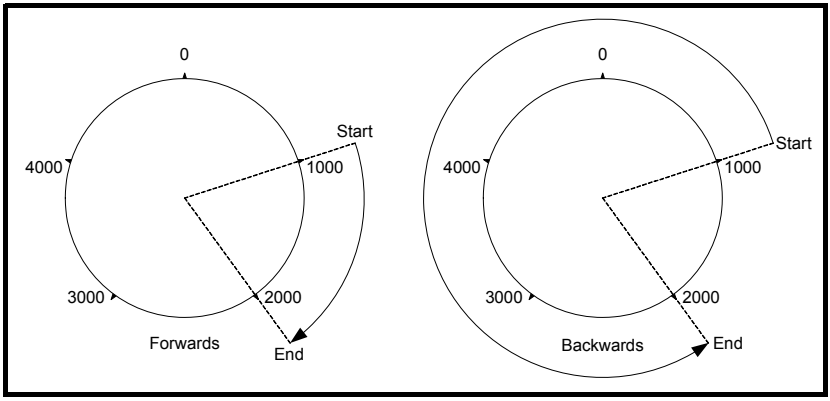
See Table 10.5 *Status% Output Table* on page 151.

10.4.7 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 SI-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 600 rpm (600 m/min, or 163840 units per second) and motion engine period of 1ms, the user must not set a window shorter than 3 ms, or 492 position units, which equates to approximately 30 mm. At lower speeds the minimum window distance will be proportionally smaller, e.g. 300 m/min defines minimum window length requirement of 15 mm.

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

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 SI-Register digital input [0].

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

Return values

Status%

See Table 10.5 *Status% Output Table* on page 151.

10.4.8 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 SI-Register digital input [0].

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

Return values

Status%

See Table 10.5 *Status% Output Table* on page 151.

10.4.9 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 SI-Register digital input [0].

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

Return values

Status%

See Table 10.5 *Status% Output Table* on page 151.

10.4.10 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 SI-Register digital input [0].

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

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151.

`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.

10.4.11 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 SI-Register digital input [0].

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

Return values

Status%

See Table 10.5 *Status% Output Table* on page 151.

10.4.12 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 SI-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 SI-Register digital input [0]

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

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151

`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)

10.4.13 SetRegistrationTrimTimings

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

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

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151

10.4.14 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 SI-Register digital input [0]

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

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151

`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.

10.4.15 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 SI-Register digital input [0]

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

`OutputArray%`

The array into which the start condition will be copied. See Chapter 10.4.11 *SetRegistrationStartCondition* on page 138 for details on the format of the data in OutputArray%

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151

10.4.16 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 SI-Register digital input [0]

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

`OutputArray%[]`

The array into which the start condition will be copied.

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151

`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%`

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 SI-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 SI-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} / 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.

10.4.17 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 139 ns units.

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

Arguments

ChannelNumber%

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

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

Return values

Status%

See Table 10.5 *Status% Output Table* on page 151

`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.

10.4.18 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 SI-Register digital input [0]

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

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151

`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.

10.4.19 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 SI-Register digital input [0]

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

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151

`Position%`

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

10.4.20 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 SI-Register digital input [0]

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

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151

`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

10.4.21 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 SI-Register digital input [0]

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

Return values

`Status%`

See Table 10.5 *Status% Output Table* on page 151

`Position%`

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

10.4.22 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 10.5 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.

Status %	Definition	Description
-16	Failure	REGISTRATION_ILLEGAL_MODE The specified mode is not valid, or the operation cannot be completed in the present mode.
-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 (Pr **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.

Status %	Definition	Description
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.

11 Inter-option synchronization

NOT CURRENTLY SUPPORTED

11.1 Overview

This scheme is in addition to the CTSync scheme (refer to Chapter 9 *CTSync* on page 99 for more information), although it can be used in conjunction with CTSync to synchronize modules in more than one drive.

The module when a Consumer, can run its Position Control tasks at the same rate or quicker than the Synchronization 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 Synchronization 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 synchronize to the Producer's signal.

NOTE Although only one SI-Applications module can be fitted to the drive, the inter-option synchronization can be used to synchronize the Position Control tasks in the SI-Applications module with the reception and transmission of data from a real time fieldbus interface installed on the drive.

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 Second Processor. 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 **Slotx Error** (where x is the slot number) with the run-time error code placed into Pr **81.50**
- Drive may be tripped on another **Slotx ****** 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 **81.14**. This is detailed in section 12.3, 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 Second Processor which instigates a drive trip. Remember, not all run-time errors instigate a drive trip.

Table 12-1 Drive display trip codes

Drive trip code	Fault	Description
Slotx* HF	Hardware Fault	The drive has detected that an System Integration Module is present, but is unable to communicate with it.
Slotx* Watchdog	Watchdog Timeout	Indicates a user program which has utilised the *watchdog* feature has failed to issue the WDOG command within 200 ms.
Slotx* Error	Error	Run-time trip generated by the Second Processor either due to a user DPL program error or some other event. The actual error code is placed into Pr **81.50**.
Slotx* Not Fitted	Not installed	Module was disconnected while operational, or module has crashed. This trip will also occur if a drive slot is configured for a Second Processor, but the module is not installed in the slot.
Slotx* Different	Different installed	This trip will occur when an Second Processor is installed to a slot previously occupied by another System Integration Module, or is installed to a previously unused slot.

* Where x determines the slot number. For example an error with the module in slot 3 would give a Slot3 Error trip.

12.3 Second Processor run-time error codes

If the Second Processor detects an error during operation the error code will be placed in the following parameter:

Pr 81.50	Second Processor 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:

Plus	Lite V2	Register
✓	✓	✓

Pr 81.14	Global Run-time Trip Enable		
Access	RW	**Range**	0/1
Default	0	**Update Rate**	N/A

Plus	Lite V2	Register
✓	✓	✓

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 Initialization and the program will not be started.

Table 12.1 SI-Applications Modules & motion processors error codes

Error Code	Reason	Drive trip?	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 Pr **81.17**=0 the value written will be automatically limited and no error will occur.	May	Yes	Yes
45	Invalid synchronization 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

Error Code	Reason	Drive trip?	ERROR task?	Prog Halted?
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 Pr **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 Pr **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 *Pr 81.44* on page 38 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 Pr **81.24** and network connections.	May	No	No
63	CTNet invalid node ID. Check Pr **81.23**.	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 20 mA.	Yes	Yes	Yes

Error Code	Reason	Drive trip?	ERROR task?	Prog Halted?
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 Pr **81.37** = 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 Initialization 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 SI-ApplicationsLite module.	Yes	Yes	Yes
76	Module type cannot be resolved. Module is not recognized.	Yes	No	Not Run
77	Inter-System Integration Module comms error with module in slot 1.	Yes	Yes	Yes
78	Inter-System Integration Module comms error with module in slot 2.	Yes	Yes	Yes
79	Inter-System Integration Module comms error with module in slot 3.	Yes	Yes	Yes
80	Inter-System Integration Module comms error with module unknown slot.	Yes	Yes	Yes
81	• Slot selected as the reference or feedback does not contain a position System Integration Module • Attempt to change the Reference source or the Feedback source in more than one task. APC internal error. See Pr **81.38**. This may be caused by one of the following: • CAM table too small • A change of too many CAM segments has occurred at the CAM table input • CAM is selected but size is zero • CAM absolute mode selected and Reset Index or Reset Position in segment is out of range	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 Unidrive M and Second Processor 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 Second Processor 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 Classic where the drive was not reset.

The task which 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 **10.038**.

Plus	Lite V2	Register
✓	✓	✓

12.5 Resource monitoring

The SI-Applications Plus module, SI-Applications Lite V2 and SI-Register modules, provide more realistic resource data than is available from parameter Pr **81.04** as shown below.

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.

Plus	Lite V2	Register
✓	✓	✓

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.

Plus	Lite V2	Register
✓	✓	✓

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.

Plus	Lite V2	Register
✓	✓	✓

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

Plus	Lite V2	Register
✓	✓	✓

Pr 88.07	Free Resource for Clock Task		
Access	RO	**Range**	0 to 95
Default	NA	**Update Rate**	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.

Plus	Lite V2	Register
✓	✓	✓

Pr 88.08	Clock Task Peak Resource Detect		
Access	RO	**Range**	0 to 95
Default	NA	**Update Rate**	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.

Plus	Lite V2	Register
✓	✓	✓

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

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

Pr 81.51			
Access	RO	**Range**	0 to 99
Default	N/A	**Update Rate**	N/A

The full version of the Second Processor firmware version can be read for the corresponding slot. This manual was written for Second Processors installed with V02.00.00 firmware. The table below shows how to construct the full firmware version from these values.

Table 12.2 Firmware version

Major Version	Minor Version	Firmware Version
2.00	0	V02.00.00

Plus	Lite V2	Register
✓	✓	✓

13 Migration guide

This chapter will outline the major differences between the SM-Applications on Unidrive SP and the SI-Applications on Unidrive M.

14 Quick reference

Refer to Chapter 5 *Parameters* on page 24 for full details of these parameters.

Table 14.1 Set-up parameters

Parameter	Description	Cross Ref	Range	Default
Pr **81.01**	Module option code	section 5.3.1 on page 26	0-499	N/A
Pr **81.02**	Module firmware version		0-99.99	
Pr **81.03**	DPL program status		0-3	
Pr **81.04**	Available system resource %		0-100 %	
Pr **81.05**	EIA-RS485 address		0-255	11
Pr **81.06**	EIA-RS485 Mode		0-255	1
Pr **81.07**	EIA-RS485 Baud rate		0-9	4
Pr **81.08**	EIA-RS485 Turn-around delay		0-255 ms	2 ms
Pr **81.09**	EIA-RS485 Tx Enable Delay		0-1 ms	0 ms
Pr **81.10**	DPL print routing		0-1	0
Pr **81.11**	Clock tick time (ms)		0-200 ms	0
Pr **81.12**	POS task schedule rate		0-6	0
Pr **81.13**	Enable autorun		0-1	1
Pr **81.14**	Global run-time trip enable		0-1	0
Pr **81.15**	Disable Reset on Trip Cleared		0-1	0
Pr **81.16**	Encoder Data Update Rate		0-3	0
Pr **81.17**	Enable parameter over-range trips		0-1	0
Pr **81.18**	Watchdog enable		0-1	0
Pr **81.19**	Save request		0-1	0
Pr **81.20**	Save on ‘UU’ trip		0-1	0
Pr **81.21**	Include menu 20 for save/ restore		0-1	0
Pr **81.22**	CTNet token ring ID		0-255	0
Pr **81.23**	CTNet node address		0-255	0
Pr **81.24**	CTNet baud-rate		0-3	1
Pr **81.25**	CTNet sync setup (SSFF)		0-9999	0
Pr **81.26**	CTNet Easy Mode 1 destination node (NNNSS)		0-25503	0
Pr **81.27**	CTNet Easy Mode 1 source parameter (MMPP)		0-9999	0
Pr **81.28**	CTNet Easy Mode 2 destination node (NNNSS)		0-25503	0
Pr **81.29**	CTNet Easy Mode 2 source parameter (MMPP)		0-9999	0
Pr **81.30**	CTNet Easy Mode 3 destination node (NNNSS)		0-25503	0
Pr **81.31**	CTNet Easy Mode 3 source parameter (MMPP)		0-9999	0

Parameter	Description	Cross Ref	Range	Default
Pr **81.32**	CTNet Easy Mode Slot 1 destination (MMPP)	section 5.3.1 on page 26	0-9999	0
Pr **81.33**	CTNet Easy Mode Slot 2 destination (MMPP)		0-9999	0
Pr **81.34**	CTNet Easy Mode Slot 3 destination (MMPP)		0-9999	0
Pr **81.35**	CTNet Sync EVENT task ID		0-4	0
Pr **81.36**	CTNet Diagnostic parameter		-3-32767	N/A
Pr **81.37**	Reject download if drive enabled		0-1	0
Pr **81.38**	APC Run-time trip		0-1	
Pr **81.39**	Inter-option module Drive Synchronization Status		0-3	
Pr **81.41**	Indexer Contol		0-3	
Pr **81.42**	Pass freeze through to drive		0-1	
Pr **81.43**	Freeze invert		0-1	
Pr **81.44**	Task priority level		0-255	
Pr **81.45**	User Setup Parameter 1		N/A	N/A
Pr **81.46**	User Setup Parameter 2		N/A	
Pr **81.47**	User Setup Parameter 3		N/A	
Pr **81.48**	DPL line number of error		32 bit	0
Pr **81.49**	User program ID		16 bit	
Pr **81.50**	Run-time error code		0-255	
Pr **81.51**	Minor software revision		0-99	N/A

Table 14.2 Timer unit parameters

Parameter	Description	Cross Ref	Range	Default
Pr **85.01**	Timer Unit Control Word	section 5.5 on page 41	13bit	N/A
Pr **85.02**	Timer Unit Status Word		0-3	
Pr **85.03**	Timer Unit 16-bit Timer Count		16bit	
Pr **85.04**	Timer Unit Wrap-around Limit		16bit	
Pr **85.05**	Timer Unit Timer Capture Cache		16bit	

Table 14.3 Digital I/O parameters

Parameter	Description	Cross Ref	Range	Default
Pr **86.01**	Digital Input 0	section 5.6 on page 44	0-1	N/A
Pr **86.02**	Digital Input 1		0-1	
Pr **86.03**	Digital Output 0		0-1	
Pr **86.04**	Digital Output 1		0-1	
Pr **86.05**	Digital Outputs 0 and 1		0-3	

Table 14.4 Status parameters

Parameter	Description	Cross Ref	Range	Default
Pr **88.01**	Error Code / Reset	section on page 45	0-9999	N/A
Pr **88.02**	Task In Error		0-50	
Pr **88.03**	POS Resource Monitoring		0-1	0
Pr **88.04**	Free Resource for Motion Engine Tasks		0-95	N/A
Pr **88.05**	Motion Engine Peak Resource Detect		0-95	
Pr **88.06**	CLOCK Task Resource Monitoring		0-1	0
Pr **88.07**	Free Resource for Clock Task		0-95	N/A
Pr **88.08**	Clock task peak resource detect		0-95	

Table 14.5 General parameters

Parameter	Description	Cross Ref	Range	Default
Pr **90.01**	Feedback encoder position (2^{32}/ rev)	section 5.8 on page 48	Signed 32-bit	N/A
Pr **90.02**	Feedback Encoder Revolution Count		Unsigned 16-bit	
Pr **90.03**	Reference Encoder Position (2^{32}/rev)		Signed 32-bit	
Pr **90.04**	Reference Encoder Revolution Count		Unsigned 16-bit	
Pr **90.10**	Drive Mode		Signed 16-bit	
Pr **90.11**	Drive Status and Control Word		Signed 16-bit	
Pr **90.12**	Event task schedule reason		Unsigned 16-bit	
Pr **90.13**	Event1 task schedule reason		Unsigned 16-bit	
Pr **90.14**	Event2 task schedule reason		Unsigned 16-bit	
Pr **90.15**	Event3 task schedule reason		Unsigned 16-bit	
Pr **90.18**	Feedback Encoder Freeze Flag		0/1	
Pr **90.19**	Feedback Encoder Freeze Position		Signed 32-bit	
Pr **90.20**	Feedback Encoder Freeze Turns		Unsigned 16-bit	
Pr **90.21**	Disable Drive Encoder Position Check		0/1	0

Parameter	Description	Cross Ref	Range	Default
Pr **90.22**	Drive Encoder Comms Transmit Register	section 5.8 on page 48	Unsigned 16-bit	N/A
Pr **90.23**	Drive Encoder Comms Receive Register		Unsigned 16-bit	
Pr **90.24**	Module slot number		Unsigned 8-bit	
Pr **90.25**	Feedback encoder marker position (2^{32}/rev)		Signed 32-bit	
Pr **90.26**	Feedback encoder marker turns (2^{16}/rev)		Unsigned 16-bit	
Pr **90.27**	Second Processor database version number		Unsigned 16-bit	
Pr **90.28**	Reference Encoder Freeze flag		0/1	
Pr **90.29**	Reference Encoder Freeze Position		Signed 32-bit	
Pr **90.30**	Reference Encoder Freeze Turns		Unsigned 16-bit	
Pr **90.31**	Feedback Encoder Turns and Coarse Position		Signed 32-bit	
Pr 90.32	Reference Encoder Turns and Coarse Position		Signed 32-bit	
Pr 90.33	Feedback Encoder Freeze Turns and Coarse Position		Signed 32-bit	
Pr 90.34	Reference Encoder Freeze Turns and Coarse Position		Signed 32-bit	
Pr **90.35**	Reference Encoder Marker Position (2^{32}/rev)		Signed 32-bit	N/A
Pr **90.36**	Reference Encoder Marker turns (2^{16}/rev)		Unsigned 16-bit	
Pr **90.37**	Feedback Encoder Marker Turns and Coarse Position		Signed 32-bit	
Pr **90.38**	Reference Encoder Marker Turns and Coarse Position		Signed 32-bit	
Pr **90.39**	Drive Keypad Button Status		Signed 16-bit	
Pr **90.40**	Event Task Trigger		Unsigned 16-bit	0
Pr **90.41**	Reference Encoder Marker Flag		0/1	N/A
Pr **90.42**	Feedback Encoder Marker Flag		0/1	N/A
Pr **90.43**	Reference Encoder Source		Unsigned 8-bit	N/A
Pr **90.44**	Feedback Encoder Source		Unsigned 8-bit	N/A
Pr **90.45**	Reference Marker Flag Enable		0/1	N/A
Pr **90.46**	Feedback Marker Flag Enable		0/1	N/A

Parameter	Description	Cross Ref	Range	Default
Pr **90.47**	Reference Freeze Enable	section 5.8 on page 48	0/1	N/A
Pr **90.48**	Feedback Freeze Enable		0/1	N/A
Pr **90.49**	APC Runtime Error ID		32bit	N/A

Table 14.6 Fast access parameters

Parameter	Description	Cross Ref	Range	Default
Pr **91.01**	Short-cut enable	section 5.10 on page 60	Unsigned 8-bit	0
Pr **91.02**	Speed set-point (Pr **01.21**)		Signed 32-bit	N/A
Pr **91.03**	Hard-speed reference (Pr **03.22**)		Signed 32-bit	N/A
Pr **91.04**	Torque setpoint (Pr **04.08**)		Signed 32-bit	N/A
Pr **91.05**	Full scale speed (rpm)		Signed 32-bit	1500
Pr **91.06**	Speed feedback		Signed 32-bit	N/A
Pr **91.07**	Current feedback (Pr **04.02**)		Signed 16-bit	N/A
Pr **91.08**	Drive analog input 1 value		±4000	N/A
Pr **91.09**	Drive analog input 2 value		±1023	N/A
Pr **91.10**	Drive analog input 3 value		±1023	N/A
Pr **91.11**	Drive analog output 1		±1023	N/A
Pr **91.12**	Drive analog output 2		±1023	N/A
Pr **91.16**	Drive digital inputs		Unsigned 8-bit	N/A
Pr **91.17**	Number of Valid CTSync Messages Received		Signed 32-bit	N/A
Pr **91.18**	Number of Bad CTSync Messages Received		Signed 32-bit	N/A
Pr **91.19**	Number of Missing CTSync Messages		Signed 32-bit	N/A
Pr **91.20**	CTSync Synchronization Signal Width too Short		Signed 32-bit	N/A
Pr **91.21**	Inter-option Synchronization Control		0 to 2	0
Pr **91.22**	Inter-option Synchronization Status		Unsigned 8-bit	N/A

Table 14.7 Second processor error codes

Error Code	Reason	Drive trip?	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 Pr **81.17** = 0 the value written will be automatically limited and no error will occur.	May	Yes	Yes
45	Invalid synchronization 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 Pr **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 Pr **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 *Pr 81.44* on page 38 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

Error Code	Reason	Drive trip?	ERROR task?	Prog Halted?
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 Pr **81.24** and network connections.	May	No	No
63	CTNet invalid node ID. Check Pr **81.23**.	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 Pr **81.37** = 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 initialization 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 SI-Applications Lite module.	Yes	Yes	Yes
76	Module type cannot be resolved. Module is not recognized.	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

Error Code	Reason	Drive trip?	ERROR task?	Prog Halted?
80	Inter-Option module comms error with module unknown slot.	Yes	Yes	Yes
81	• Slot selected as the reference or feedback does not contain a position option module • Attempt to change the Reference source or the Feedback source in more than one task. APC internal error. See parameter Pr **81.38**. This may be caused by one of the following: • CAM table too small • A change of too many CAM segments has occurred at the CAM table input • CAM is selected but size is zero • CAM absolute mode selected and Reset Index or Reset Position in segment is out of range	May	Yes	Yes
82	Communications to drive faulty.	May	Yes	Yes

Table 14.8 Terminals

Terminal	Function	Description
1	0V SC	0V connection for EIA-RS485 port
2	/RX	EIA-RS485 Receive line (negative). Incoming.
3	RX	EIA-RS485 Receive line (positive). Incoming.
4	/TX	EIA-RS485 Transmit line (negative). Outgoing.
5	TX	EIA-RS485 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

Index

0478-0009-01