

## **SM-APPLICATIONS LITE** **Solutions module for UNIDRIVE SP**

User guide

## **General Information**

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

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

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

## **Drive software version**

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

If there is any doubt, contact LEROY-SOMER.

Hardware: SM-Applications Lite issue 01.00

Firmware: V01.03.03

---

# Contents

---

<b>1</b>	<b>Safety Information</b>	<b>5</b>
1.1	Warnings, Cautions and Notes	5
1.2	Electrical Safety - General Warning	5
1.3	System Design and Safety of Personnel	5
1.4	Environmental Limits	6
1.5	Compliance with Regulations	6
1.6	Motor	6
1.7	Adjusting Parameters	6
<b>2</b>	<b>Introduction</b>	<b>7</b>
2.1	Introduction	7
2.2	SM-Applications Lite module for Unidrive SP	7
2.3	Specifications	8
2.4	PC development software	8
2.5	User knowledge	8
<b>3</b>	<b>Installation</b>	<b>9</b>
3.1	Mechanical installation	9
3.2	SMARTCARD installation	9
<b>4</b>	<b>Getting started</b>	<b>10</b>
4.1	Connecting the PC to the SM-Applications Lite module	11
4.2	Using SYPT Workbench	11
4.3	Using SYPTLite	15
<b>5</b>	<b>Parameters</b>	<b>17</b>
5.1	Overview	17
5.2	Saving Parameters	17
5.3	Configuration Parameters	18
5.4	Menus 70-75 - PLC Registers	24
5.5	Menu 88 - Status Parameters	24
5.6	Menu 90 - General Parameters	25
5.7	Menu 91 - Fast Access Parameters	32
5.8	Menus 18,19 - Application Parameters	36
5.9	Menu 20 - Application Menu	37
<b>6</b>	<b>Communications</b>	<b>38</b>
6.1	SM-Applications Lite mapping parameters	38

<b>7</b>	<b>DPL programming</b>	<b>40</b>
7.1	Program header	40
7.2	Tasks	41
7.3	Variables	43
7.4	Parameters	45
7.5	Operators	46
7.6	Basic DPL commands	47
7.7	User defined function blocks	50
<b>8</b>	<b>SM-Applications Lite Freeze and Marker</b>	<b>53</b>
8.1	Freeze input	53
8.2	Marker pulse	54
<b>9</b>	<b>Inter-Option Module Synchronisation</b>	<b>56</b>
9.1	Overview	56
9.2	Inter-Option Synchronisation example	56
9.3	Position control tasks	59
<b>10</b>	<b>Diagnostics</b>	<b>60</b>
10.1	Run-time Errors	60
10.2	Unidrive SP Trip Display Codes	60
10.3	SM-Applications Lite Run-time Error Codes	61
10.4	Handling Run-Time Errors with the ERROR task	62
10.5	Support	63
<b>11</b>	<b>Migration Guide</b>	<b>64</b>
11.1	Drive Parameter Differences	64
11.2	UD70 Parameters	64
11.3	General Features	65
11.4	SM-Applications Porting Aid	67
<b>12</b>	<b>Quick Reference</b>	<b>68</b>

---

# 1 Safety Information

---

## 1.1 Warnings, Cautions and Notes



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



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

### 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 and maintenance must be carried out by personnel who have the necessary training and experience. They must read this safety information and this User Guide carefully.

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

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

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

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

<sup>1</sup>Independent approval by BIA has been given for sizes 1 to 3.

## 1.4 Environmental Limits

Instructions in the *Unidrive SP 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 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 earth (ground) connections.

The *Unidrive SP User 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:

98/37/EC: Safety of machinery.

89/336/EEC: Electromagnetic Compatibility.

## 1.6 Motor

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

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

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

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

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

## 1.7 Adjusting Parameters

Some parameters have a profound effect on the operation of the drive. They must not be altered without careful consideration of the impact on the controlled system.

Measures must be taken to prevent unwanted changes due to error or tampering.

---

## 2 Introduction

---

**NOTE**

Unidrive SP parameters are denoted in this manual by “#MM.PP”, where MM refers to the menu number, and PP refers to the parameter number within that menu. Please refer to the Unidrive SP manual for full parameter definitions.

### 2.1 Introduction

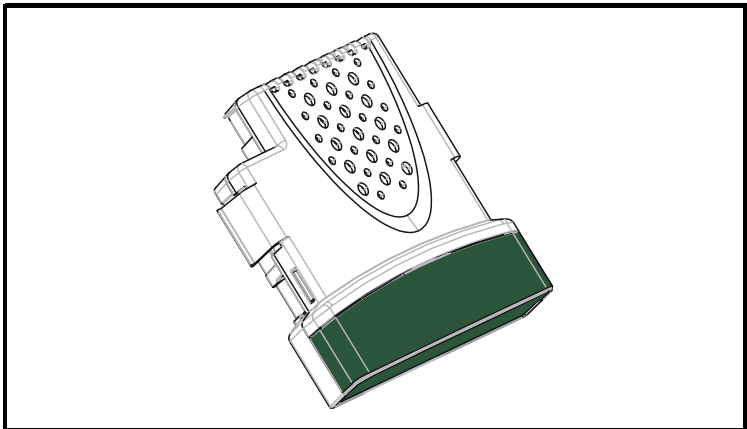
Modern variable speed drives such as the Unidrive SP offer a multitude of in-built features such as ramp control, PID loops, simple position control, etc. However this functionality is limited. The drive can only do so many things and when it comes to controlling more complex applications, users often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

However the flexibility of the Unidrive SP can be substantially increased by using an SM-Applications Lite module. This module is a second processor for the drive and allows the user to utilise existing, or write their own, application-specific software.

### 2.2 SM-Applications Lite module for Unidrive SP

The SM-Applications Lite module for Unidrive SP is an option module that can be fitted to any one of the three expansion slots in the Unidrive SP.

**Figure 2-1 Unidrive SP option module**



The SM-Applications Lite module is powered from the Unidrive SP internal power supply.

## 2.3 Specifications

- High speed dedicated microprocessor
- SYPTLite 10kb executable memory
- SYPT 100kb executable memory, 20kb user memory
- Dual-port RAM interface for communicating with the Unidrive SP and other option modules
- Task based programming system allowing for real-time control of drive and process.

## 2.4 PC development software

Application programs for the SM-Applications Lite module may be developed by the user using the SYPT Workbench or SYPTLite programming packages.

SYPTLite offers Ladder programming for quick creation of simple applications.

SYPT Workbench offers the following tools to help in developing more powerful solutions:

- Ladder and function block programming
- Native DPL programming
- Watch window for monitoring any drive or option parameter as well as user program variables
- Single-stepping and breakpoint debugging facilities

With both software packages you must connect to the SM-Applications Lite via a direct connection to the RS485 port on the front of the Unidrive SP.

Both software packages run under Microsoft Windows™ 98/ME/NT4/2000/XP.

## 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 (LD) and function block (FBD) facilities of the SYPT Workbench/SYPTLite 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

---

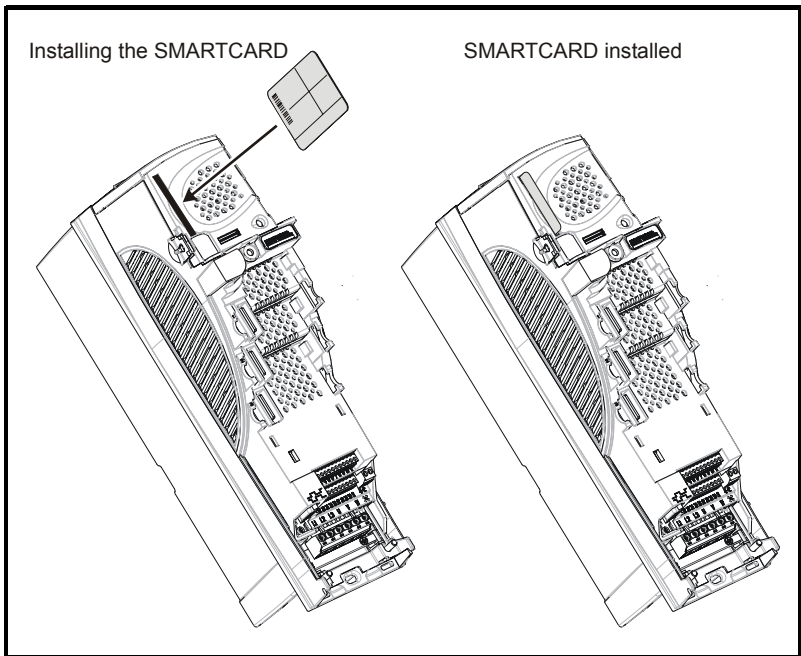
### 3.1 Mechanical installation

Please refer to the installation sheet that comes with the SM-Applications Lite module for details of installing the module into the Unidrive SP.

### 3.2 SMARTCARD installation

The Unidrive SP can be fitted with a Smartcard (see Figure 3-1) which allows the user to read and write data. This is useful for storing parameter sets and other data values. For more information on using the Smartcard from the keypad, refer to the Unidrive SP User Guide. Refer to section 7.6.1 *New commands for SM-Applications Lite module* for information on using DPL commands with the Smartcard.

**Figure 3-1** Installing the SMARTCARD



**Figure 3-2**

## 4 Getting started

This chapter describes the basics of a user program using the SM-Applications Lite module and some of the aspects of using the SYPT Workbench and SYPTLite.

Unidrive SP has 3 slots available for option modules, and each slot has a dedicated menu of configuration parameters.

Figure 4-1 Unidrive SP slot arrangement

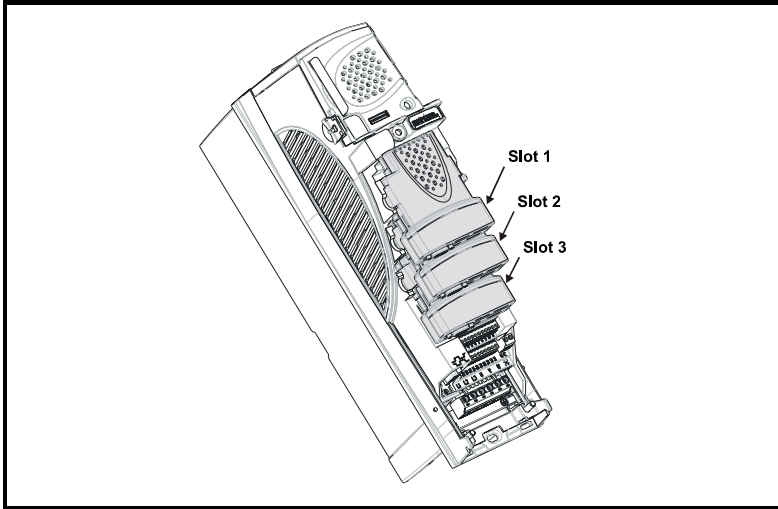


Table 4.1 Slot configuration menu

Slot	Menu
1	#15.PP
2	#16.PP
3	#17.PP

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

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

When the SM-Applications Lite module is fitted, the module identification parameter #81.01 will show the value 302. The combination of parameters #81.02 and #81.51 provides the firmware version of the module. This User Guide is written for firmware version V01.03.03.

## 4.1 Connecting the PC to the SM-Applications Lite module

Using a special RS232 to RS485 converter you can connect the PC to the RJ45 serial port on the front of the drive. A special pre-made lead is available from LEROY-SOMER for this purpose - this same lead is used with other LEROY-SOMER products that use a RJ45 RS485 connector such as the Digidrive SE.

The RJ45 connector is located under a small flap on the front of the Unidrive SP just below the keypad. The pin-outs of this connector are described in the Unidrive SP User Guide.

With this type of connection you can only control the drive and SM-Applications Lite module that the PC is connected to.

**Figure 4-2 Communications cable**



## 4.2 Using SYPT Workbench

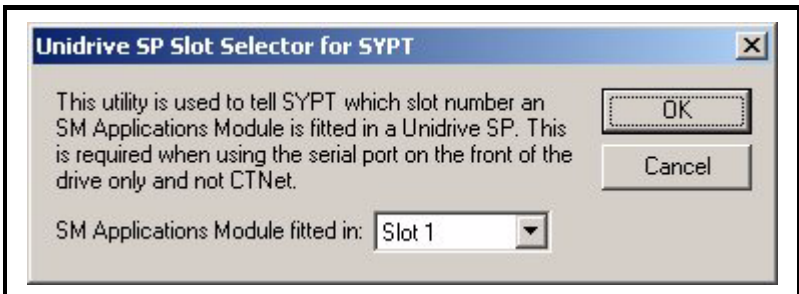
The SYPT Workbench program, as used by the UD70 product for UMV 4301, provides one of the development platforms for SM-Applications Lite module. There is also SYPTLite, which is covered in section 4.3 *Using SYPTLite*

There are certain caveats with using SYPT Workbench with the SM-Applications Lite that will be covered later.

### 4.2.1 SM-Applications Lite slot selection

When using SYPT Workbench with the RS232/485 connection you must inform SYPT as to which slot the SM-Applications Lite module is fitted in. The *Slot Selector* program (available from Start->Programs->SYPT Workbench menu) is used to do this.

**Figure 4-3 Slot select program**



## 4.2.2 Configuring communications in SYPT

Before attempting to go *on-line* to the SM-Applications Lite module you must set SYPT up to use the correct communications protocol:

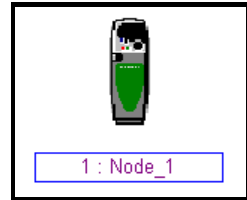
1. In the SYPT Configuration Editor select **Communications** from the Options menu
2. If connecting via RS232/485 to the front of the drive select **CT-RTU** as the protocol and choose the appropriate RS232 COM port. Also ensure drive parameters #11.24 and #11.25 are set to **rtu** and **19200** respectively (these are the default values). At present SYPT can only communicate at 19200 baud with this protocol.
3. Press OK.

## 4.2.3 Creating a Unidrive SP node in SYPT

A Unidrive SP node is created in the same way as other node (drive) types:

1. Insert a new node by selecting **Node** from the Insert menu.
2. Double-click on the name of the new node (or select Edit->Node properties).
3. From the Target Type drop down, select **SMAPPL** (the drive type will automatically change to Unidrive SP).
4. Press OK.

The new node will now be a Unidrive SP.



## 4.2.4 Porting UD70 programs to the SM-Applications Lite module

If you intend to convert programs from the UD70 platform for UMV 4301 to the Unidrive SP please read Chapter 11 *Migration Guide* for important information.

## 4.2.5 DPL programming basics

With SYPT Workbench the SM-Applications Lite module can be programmed using a mixture of ladder diagrams (LD), function block diagrams (FBD) and DPL code (Drive Programming Language). Collectively they are known as a *DPL Program*.

### NOTE

Only the ladder programming is available with SYPT Lite.

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 dialog box in SYPT.
- Program Body - comprised of *task* sections containing LD, FBD and DPL sections. This is created in the DPL Editor within SYPT.

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

## 4.2.6 Function block library

The SYPT Workbench 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).

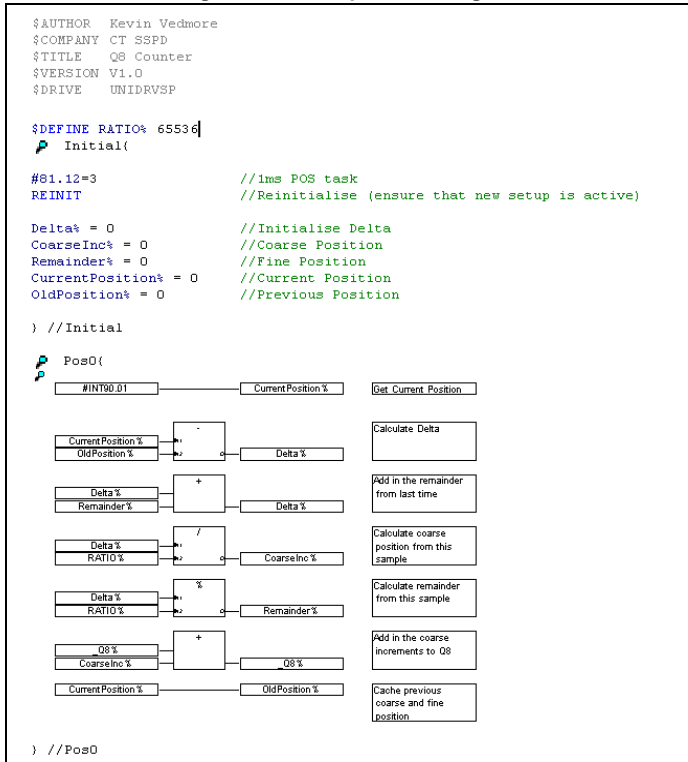
The functions in the FBL are documented in the on-line help.

You can also create your own function blocks within your program. So if you've created a new jazzy 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* and the on-line help for information.

## 4.2.7 Program example

Shown below is an example of a DPL program:

Figure 4-4 Example DPL Program



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

This example shows the basic concepts of accessing parameters and using mathematical functions. It may be useful to people migrating from the UD70 platform for UMV 4301 since it will show how to re-create the `_Q8%` accumulative encoder position value as was available on that product.

Running through the program we have four distinct sections:

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

#### 4.2.8 Header section

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

#### 4.2.9 Initial task

As explained later in section 7.2 *Tasks*, this is a *task* which is executed when the SM-Applications Lite is first powered up or is reset. In this task are some DPL statements that initialise some integer variables (denoted by a trailing % symbol) to zero, and one variable to the value of a special parameter `#90.01`.

This `#90.01` parameter provides the current encoder position scaled to  $2^{31}/\text{rev}$  regardless of the encoder type fitted.

#### 4.2.10 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*.

#### NOTE

If you wish to create and try this program yourself and you have not used the SYPT Workbench or SYPTLite software before then it is advised to read through the remainder of this chapter first, then read the appropriate documentation of the software you are using.

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

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

### 4.2.11 SYPT Workbench caveats

The following are known issues in using the SYPT Workbench with the SM-Applications Lite module:

- **Slot selection**  
If you are connecting to the Unidrive SP by means of the front RJ45 serial connector, SYPT by default will be expecting the SM-Applications Lite module to be inserted into SLOT 1 of the drive. However you can change this to any of the other slots by using the SLOT SELECT utility provided. (Get to it via the Start Menu->Programs->SYPT Workbench->Slot Selector.)
- **Breakpoints in POS0/1 tasks**  
If you set a breakpoint in either of these tasks, the breakpoint will occur but the flag will not be shown and it is only possible to clear the breakpoint by resetting the option module.
- **EVENTx tasks**  
It is not possible to single-step or place breakpoints in any of the EVENT tasks
- **User defined function blocks (UDFB's)**  
The SM-Applications Lite module is much more flexible with regard the input and output arguments of UDFB's (more inputs and outputs are available and no alignment restrictions). However SYPT still imposes the UD70 restrictions so it's not possible to take advantage of the new flexibility at this time.

## 4.3 Using SYPTLite

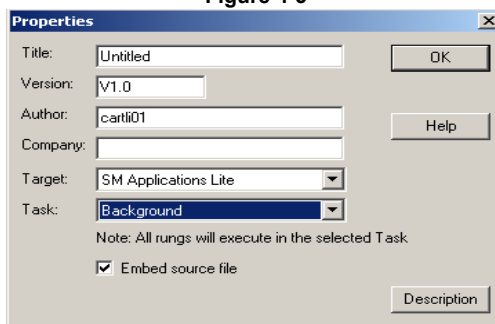
The SYPTLite program provides an alternative development platform to the SYPT Workbench for the Applications-Lite module.

SYPTLite is the 'entrance' level programming tool which is delivered free-of-charge with the Unidrive SP and can be used straight out of the box for programming in quick ladder logic.

Connectivity to the PC is the same as mentioned in section 4.1 *Connecting the PC to the SM-Applications Lite module*.

When using SYPTLite there is a choice of 2 'targets' when you create a new ladder diagram, Unidrive SP Onboard or SM Applications Lite.

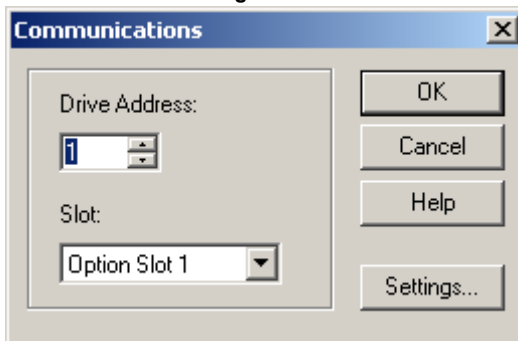
Figure 4-5



On choosing SM-Applications Lite, you have the option of 3 tasks to choose from. Background, Clock or Initial. You will only be able to use **one** of these tasks for the whole program.

The 'Slot Selector' is not needed in SYPTLite. In the 'Communications' settings from the drop-down menu the following box will appear.

**Figure 4-6**



All of the programming in SYPTLite is in ladder and you are limited to 10kb of executable memory.

For further information see Help in SYPTLite.

---

# 5 Parameters

---

## 5.1 Overview

The SM-Applications Lite module contains two parameter databases:

- The Unidrive SP database  
This contains the entire drive parameter set. The SM-Applications Lite caches this database in its own non-volatile *flash* memory. At power-up the module will check to see if this cache matches that of the drive. If it doesn't the database will be loaded from the drive, during which time the word "Loading" will appear for a few seconds on the drive display. This will not occur again unless the SM-Applications Lite is moved to a different drive with different firmware or the drive firmware is updated.
- The SM-Applications Lite module 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 SM-Applications Lite Parameters

The parameters that are saved to the SM-Applications Lite module when performing the actions shown below are:

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

To save the parameters on demand:

1. Set #81.19 to 1
2. Press the reset button  
Parameter #81.19 will reset to zero automatically and the module and drive will be reset.

To save the parameters on Under Voltage (UU):

- Set #81.20 to 1.

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 #81.21 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 #81.21 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 thru 14, 18, 19 and 21.
- Menus 15, 16 and 17, if a module is present in the relevant slot.

To save the drive parameters:

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

## 5.3 Configuration Parameters

The basic configuration (or setup) parameters are held in the appropriate menu for the slot where the SM-Applications Lite module is fitted.

Slot	Menu
1	15
2	16
3	17

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



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

To reset the SM-Applications Lite 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 #81.xx. When setting parameters directly on the drive keypad use the appropriate menu 15, 16 or 17 instead.

### NOTE

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



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

### 5.3.1 Parameter Descriptions

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

The Module Code indicates the type of module that is fitted in the corresponding slot. For a SM-Applications Lite module the code is **302**.

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

Specifies the major revision number of the operating system of the SM-Applications Lite module. Use in conjunction with #81.51 to form the complete version number.

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

Provides the run status of the user DPL program in the SM-Applications Lite module. 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

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

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

#81.11	Clock Task Scheduling (ms)		
Access	RW	Range	0 to 200ms
Default	10ms	Update Rate	Initialisation

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

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

#81.12	POS task scheduling rate		
Access	RW	Range	0 to 6
Default	0	Update Rate	Initialisation

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

Display	Value	Description
diSAbled	0	Disabled
0.25	1	250 $\mu$ s
0.5	2	500 $\mu$ s
1	3	1ms
2	4	2ms
4	5	4ms
8	6	8ms

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

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

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

#81.14	Global Run-time Trip Enable		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	0	<b>Update Rate</b>	Initialisation

Setting this parameter to 1 will cause the Unidrive SP to trip when certain run-time errors occur within the SM-Applications Lite module / user DPL program.

For more information, see section 10.1 *Run-time Errors*.

#81.15	Disable Reset on Trip Cleared		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	0	<b>Update Rate</b>	Initialisation

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

#81.16	Encoder Data Update Rate		
<b>Access</b>	RW	<b>Range</b>	0-3
<b>Default</b>	0	<b>Update Rate</b>	Initialisation

When this parameter is 0 the APC data and menu 90 encoder parameters are updated every 250 $\mu$ s.

When this parameter is 1 the APC data and menu 90 encoder parameters are updated immediately prior to every POS task.

When this parameter is 2 the APC data and menu 90 encoder parameters are updated immediately prior to every CLOCK task.

When 3 the APC data and menu 90 encoder parameters are never updated. If these are never updated, more processor resource will become free.

#81.17	Enable Parameter Over-range Trips		
Access	RW	Range	0/1
Default	0	Update Rate	Initialisation

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

#81.18	Watchdog Enable		
Access	RW	Range	0/1
Default	0	Update Rate	Initialisation

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

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

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

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

#81.20	Enable "UU trip" Save		
Access	RW	Range	0/1
Default	0	Update Rate	Immediate

Setting this parameter to 1 signals that all non-volatile data of the SM-Applications Lite module will be automatically saved upon a power-down of the drive.

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

#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 (#x.19=1) or power-down save (#x.20=1). If menu 20 is to be restored on power-up the user must ensure that this parameter is saved in the drive before powering down.

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

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

#81.37	Reject Download if Drive Enabled		
Access	RW	Range	0/1
Default	0	Update Rate	Initialisation

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

#81.38	APC Run-time trip		
Access	RW	Range	0/1
Default	0	Update Rate	Initialisation

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

#81.39	Inter-SM-Applications Lite Drive Sync Status		
Access	RW	Range	0/1
Default	0	Update Rate	NA

This parameter displays the current module's synchronisation status. It will indicate whether or not the module is the Synchronisation master.:

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

#81.43	Freeze Invert		
Access	RW	Range	0/1
Default	0	Update Rate	Initialisation

When this parameter is set to zero the SM-Applications Lite will freeze data when it sees a rising edge of a freeze pulse from the drive. When it is set to 1 the SM-Applications Lite will freeze data when it sees a falling edge of a freeze pulse from the drive. For further information on the Freeze Input refer to Chapter 8 *SM-Applications Lite Freeze and Marker*.

#81.44	Task Priority Level		
Access	RW	Range	0 to 255
Default	0	Update Rate	Initialisation

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

Bit	Value	Meaning
1	0	Inter-option communication task priority is higher than the POS tasks.
	1	Inter option communication task priority lower than the POS tasks.

#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).

#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.

#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 section 10.1 *Run-time Errors* for further information.

#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 SM-Applications Lite module. Use in conjunction with #81.02 to form the complete version number.

## 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]	100	General purpose. Saveable.
72	RW	_Rx%, _R%[x]	100	Used for outgoing CTNet cyclic data links. Non-saveable.
73	RW	_Sx%, _S%[x]	100	Used to incoming CTNet cyclic data links. Non-saveable
74	RW	_Tx%, _T%[x]	100	General purpose. Saveable.
75	RW	_Ux%, _U%[x]	100	General purpose. Saveable.

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

e.g. **#72.01=1** will do the same as **\_R01%=1**, **#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* 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.

## 5.5 Menu 88 - Status Parameters

#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 run-time error code the same as #81.50 (note - it will not return drive trip codes). The parameter is cleared to zero on reset and when the user program execution is started.

When the parameter is written to with a value of 1070 the SM-Applications Lite module will initiate a warm-restart of the drive and any other options. This can be used to restart the user program (providing auto-run #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 parameter #88.01 will result in any drive trip being automatically cleared as well as resetting all fitted options in the Unidrive SP. This behaviour is different to the UD70 product on UMV 4301 where the drive was not reset.

#88.02	Task in Error		
<b>Access</b>	RO	<b>Range</b>	0 to 50
<b>Default</b>	N/A	<b>Update Rate</b>	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

A value of zero will be returned if there is no error condition.

For more information on these parameters refer to Chapter 10 *Diagnostics* .

## 5.6 Menu 90 - General Parameters

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

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

#90.01	Feedback Encoder Position ( $2^{32}/rev$ )		
<b>Access</b>	RO	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	See #81.16

Contains the feedback encoder position.

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

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



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

#90.02	Feedback Encoder Revolution Count		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	See #81.16

Contains the feedback encoder revolution count.

#90.03	Reference Encoder Position ( $2^{32}/\text{rev}$ )		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	See #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.

#90.04	Reference Encoder Revolution Count		
Access	RO	Range	Unsigned 16-bit
Default	N/A	Update Rate	See #81.16

Contains the reference encoder revolution count.

#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 Unidrive SP is in. It is recommended that this parameter is used instead of #11.31 or #0.48 since those parameters indicate the requested, not the actual, mode.

The values are defined as follows:

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

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

#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 parameter #10.40).

**Table 5.1 Control Word**

Bit	Description
b15	If set, the value of #1.46 is set from b6
b14	If set, the value of #1.45 is set from b5
b13	Sets the value of #18.33 (Application menu 1, bit 3)
b12	If set, the value of #6.32 is set from b3
b11	If set, the value of #6.31 is set from b2
b10	If set, the value of #6.30 is set from b1
b9	If set, the value of #6.15 is set from b0
b8	Sets the value of #18.32 (Application menu 1, bit 2)
b7	Sets the value of #18.31 (Application menu 1, bit 1)
b6	Sets the value of #1.46 (Preset select bit 1)
b5	Sets the value of #1.45 (Preset select bit 0)
b4	User trip. Trips drive immediately if set.
b3	Sets the value of #6.32 (Sequencing bit 2: Run Reverse)
b2	Sets the value of #6.31 (Sequencing bit 1: Jog)
b1	Sets the value of #6.30 (Sequencing bit 0: Run forward)
b0	Sets the value of #6.15 (Drive enable)

**Table 5.2 Status Word**

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

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

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

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

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

These four parameters 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
0-1	Slot triggering task 0 = Local slot 1 = Slot 1 2 = Slot 2 3 = Slot 3
2-7	Reason for trigger. 0-31 = Other option module initiated 32 = CTNet Sync 33 = Timer Unit 34-63 = User-defined reason via the DPL command SCHEDULEEVENT.

#90.18	Feedback Encoder Freeze Flag		
Access	RW	Range	0/1
Default	N/A	Update Rate	250µs

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

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

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

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

<b>#90.21</b>	<b>Disable Drive Encoder Position Check</b>		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	0	<b>Update Rate</b>	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.

<b>#90.22</b>	<b>Drive Encoder Comms Transmit Register</b>		
<b>Access</b>	RW	<b>Range</b>	Unsigned 16-bit
<b>Default</b>	N/A	<b>Update Rate</b>	Immediate

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

<b>#90.23</b>	<b>Drive Encoder Comms Receive Register</b>		
<b>Access</b>	RW	<b>Range</b>	Unsigned 16-bit
<b>Default</b>	N/A	<b>Update Rate</b>	Immediate

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

<b>#90.24</b>	<b>SM-Applications Lite slot number</b>		
<b>Access</b>	RO	<b>Range</b>	Unsigned 8-bit
<b>Default</b>	N/A	<b>Update Rate</b>	Initialisation

This parameter reports the slot number into which the SM-Applications Lite module is fitted.

<b>#90.25</b>	<b>Feedback encoder marker position (<math>2^{32}/\text{rev}</math>)</b>		
<b>Access</b>	RO	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	See #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.

<b>#90.26</b>	<b>Feedback encoder marker turns (<math>2^{16}/\text{rev}</math>)</b>		
<b>Access</b>	RO	<b>Range</b>	Unsigned 16-bit
<b>Default</b>	N/A	<b>Update Rate</b>	See #81.16

This parameter gives the feedback encoder marker revolution count.

<b>#90.27</b>	<b>SM-Applications Lite database version number</b>		
<b>Access</b>	RO	<b>Range</b>	Unsigned 16-bit
<b>Default</b>	N/A	<b>Update Rate</b>	Initialisation

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

<b>#90.28</b>	<b>Reference Encoder Freeze flag</b>		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	N/A	<b>Update Rate</b>	250 $\mu$ s

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

<b>#90.29</b>	<b>Reference Encoder Freeze Position</b>		
<b>Access</b>	RO	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	250 $\mu$ s

<b>#90.30</b>	<b>Reference Encoder Freeze Turns</b>		
<b>Access</b>	RO	<b>Range</b>	Unsigned 16-bit
<b>Default</b>	N/A	<b>Update Rate</b>	250 $\mu$ s

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

<b>#90.31</b>	<b>Feedback Encoder Turns and Coarse Position</b>		
<b>Access</b>	RO	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	See #81.16

<b>#90.32</b>	<b>Reference Encoder Turns and Coarse Position</b>		
<b>Access</b>	RO	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	See #81.16

These 2 parameters store the 16-bit turns in the upper word and 16-bit position in the lower word, of the feedback (#90.31) and reference (#90.32) encoders.

<b>#90.33</b>	<b>Feedback Encoder Freeze Turns and Coarse Position</b>		
<b>Access</b>	RO	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	250 $\mu$ s

<b>#90.34</b>	<b>Reference Encoder Freeze Turns and Coarse Position</b>		
<b>Access</b>	RO	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	250 $\mu$ s

These 2 parameters 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.

<b>#90.35</b>	<b>Reference Encoder Marker Position (<math>2^{32}/\text{rev}</math>)</b>		
<b>Access</b>	RO	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	See #81.16

This parameter stores the reference encoder position at the time the marker pulse was activated.

<b>#90.36</b>	<b>Reference Encoder Marker turns (<math>2^{16}/\text{rev}</math>)</b>		
<b>Access</b>	RO	<b>Range</b>	Unsigned 16-bit
<b>Default</b>	N/A	<b>Update Rate</b>	See #81.16

This parameter stores the reference encoder revolution count at the time the marker pulse was activated.

<b>#90.37</b>	<b>Feedback Encoder Marker Turns and Coarse Position</b>		
<b>Access</b>	RO	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	See #81.16

<b>#90.38</b>	<b>Reference Encoder Marker Turns and Coarse Position</b>		
<b>Access</b>	RO	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	See #81.16

These 2 parameters store the 16-bit turns in the upper word and 16-bit position in the lower word, of the feedback (#90.37) and reference (#90.38) encoders at the time the marker pulse was activated.

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

<b>#90.40</b>	<b>Event Task Trigger</b>		
<b>Access</b>	RW	<b>Range</b>	Unsigned 16-bit
<b>Default</b>	0	<b>Update Rate</b>	Immediate

Upon setting this parameter to a value it will execute one of the SM-Applications Lite Event tasks.

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

<b>#90.41</b>	<b>Reference Encoder Marker Flag</b>		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	N/A	<b>Update Rate</b>	See #81.16

<b>#90.42</b>	<b>Feedback Encoder Marker Flag</b>		
<b>Access</b>	RW	<b>Range</b>	0/1
<b>Default</b>	N/A	<b>Update Rate</b>	See #81.16

These 2 parameters are set to 1 if a relevant encoder marker pulse is activated, but only if the marker flag enable parameters have been set (parameters #90.45 and #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.

#90.43	Reference Encoder Source		
Access	RW	Range	Unsigned 8-bit
Default	N/A	Update Rate	Immediate

#90.44	Feedback Encoder Source		
Access	RW	Range	Unsigned 8-bit
Default	N/A	Update Rate	Immediate

These 2 parameters define the source for the reference and feedback data. See the table below for the valid sources.

Value	Description
0	Drive encoder
1	Slot 1
2	Slot 2
3	Slot 3
4	User program
5	Unconfigured

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

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

These 2 parameters must be set to 1 allow the marker flags (parameters #90.41 and #90.42) to be set when the marker pulse is activated.

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

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

These 2 parameters must be set to 1 allow the freeze flags (parameters #90.18 and #90.28) to be set when the freeze input is activated.

## 5.7 Menu 91 - Fast Access Parameters

The parameters in this menu are SM-Applications Lite virtual parameters which provide a faster update rate or enhanced resolution than drive parameters.

#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	#91.02
1	Hard-speed reference shortcut enable	#91.03
2	Torque reference shortcut enable	#91.04
3	Analogue Output 1	#91.11
4	Analogue Output 2	#91.12
5	Reserved	-
6	Reserved	-
7	Reserved	-

#91.02	Speed set-point (#1.21)		
<b>Access</b>	RW	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	250µs

Sets the speed reference in units of **0.001RPM**. This value is mirrored in drive parameter #1.21 (preset speed 1), therefore in order to control the drive speed with this parameter ensure preset speed 1 is selected on the drive (#1.14=3, #1.15=1).

Ensure bit 0 of #91.01 is set and the full-scale speed in #91.05 is set accordingly when using this parameter. Note that this parameter is valid in Closed Loop Vector and Servo modes only.

#91.03	Hard-speed reference (#3.22)		
<b>Access</b>	RW	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	250µs

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

Ensure bit 1 of #91.01 is set and the full-scale speed in #91.05 is set accordingly when using this parameter. Note that this parameter is valid only in Closed Loop Vector and Servo modes only.

#91.04	Torque setpoint (#4.08)		
<b>Access</b>	RW	<b>Range</b>	Signed 32-bit
<b>Default</b>	N/A	<b>Update Rate</b>	250µs

Specifies the torque setpoint (drive parameter #4.08) in units of 0.01%.

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

#91.05	Full scale speed (RPM)		
<b>Access</b>	RW	<b>Range</b>	Signed 32-bit
<b>Default</b>	1500	<b>Update Rate</b>	N/A

Set this to the maximum (absolute) speed that will ever be written to with #91.02 or #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 #91.02 or #91.03 greater than the RPM value specified in #91.05 will result in the value being limited or a value overrange run-time error.

#91.06	Speed feedback		
Access	RO	Range	Signed 32-bit
Default	N/A	Update Rate	250 $\mu$ s

This parameter returns the value of the Unidrive SP speed feedback in units of 0.01RPM in closed loop modes. However, if a low resolution encoder is used there may be some jitter at low speed. For example, at 10rpm with a 1024ppr encoder this parameter may jump between 0 and 14.65rpm. This is similar to the Unidrive SP parameter #3.02.

#91.07	Current feedback (#4.02)		
Access	RO	Range	Signed 16-bit
Default	N/A	Update Rate	250 $\mu$ s

This value is taken from parameter #4.02 and is in 0.01A units (i.e. 150=1.5 Amps)

#91.08	Drive analog input 1 value		
Access	RO	Range	$\pm$ 4000
Default	N/A	Update Rate	250 $\mu$ s

This value will be taken from the drive's analog input 1 and is scaled for  $\pm$ 4000 to represent the +/- full scale signal at the input. Refer to the drive user guide for information on the sampling rate of analogue inputs.

#91.09	Drive analog Input 2 value		
Access	RO	Range	$\pm$ 1000
Default	N/A	Update Rate	250 $\mu$ s

This value will be taken from the drive's analog input 2 and is scaled for  $\pm$ 1000 to represent the +/- full scale signal at the input. Refer to the drive user guide for information on the sampling rate of analogue inputs.

#91.10	Drive analog input 3 value		
Access	RO	Range	$\pm$ 1000
Default	N/A	Update Rate	250 $\mu$ s

This value will be taken from the drive's analog input 3 and is scaled for  $\pm$ 1000 to represent the +/- full scale signal at the input. Refer to the drive user guide for information on the sampling rate of analogue inputs.

#91.11	Drive analog output 1		
Access	RW	Range	$\pm$ 1023
Default	N/A	Update Rate	NA

This parameter sets the value of analog output 1. Refer to parameter #91.01 for information on enabling this parameter.

#91.12	Drive analog output 2		
Access	RW	Range	$\pm$ 1023
Default	N/A	Update Rate	NA

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

#91.16	Drive digital inputs		
<b>Access</b>	RO	<b>Range</b>	Unsigned 8-bit
<b>Default</b>	N/A	<b>Update Rate</b>	250µs

This parameter is similar to drive parameter #8.20 in providing the status of 7 digital inputs in one single parameter. Logic polarity and inversions are taken into account.

The bits are assigned as follows:

Bit	Digital Input
0	F1
1	F2
2	F3
3	F4
4	F5

Bit	Digital Input
5	F6
6	Enable
7	Reserved - Read as zero

#91.21	Inter-option Synchronisation Control		
<b>Access</b>	RW	<b>Range</b>	0 to 2
<b>Default</b>	0	<b>Update Rate</b>	Immediate

This parameter allows the user to set up the SM-Applications Lite module in the Inter-Option module Synchronisation scheme. For more information refer to Chapter 9 *Inter-Option Module Synchronisation*

Bit	Description
0	Set this bit for the SM-Applications Lite to participate in the Inter-Option Module Synchronisation scheme as a Producer. Refer to section 9.1 <i>Overview</i> for details of the term Producer.
2	Set this bit for the SM-Applications Lite to participate in the Inter-Option Module Synchronisation scheme as a Consumer. Refer to section 9.1 <i>Overview</i> for details of the term Consumer.

#91.22	Inter-option Synchronisation Status		
<b>Access</b>	RO	<b>Range</b>	Unsigned 8-bit
<b>Default</b>	N/A	<b>Update Rate</b>	Immediate

This parameter shows the status of the SM-Applications Lite in the Inter-option module synchronisation scheme. For more information refer to Chapter 9 *Inter-Option Module Synchronisation*.

Bit	Meaning	Description
0	Requested Inter-Module Synchronisation Role	This is identical to bit 0 of the Inter-Option Module Synchronisation Control Parameter as described above.
1	Requested Inter-Module Synchronisation Role	This is identical to bit 1 of the Inter-Option Module Synchronisation Control Parameter as described above.

Bit	Meaning	Description
2	Inter-Module Synchronisation Role Achieved	This bit indicates that for a module attempting to become inter-module synchronisation Producer it has achieved that role. In the event of more than one module on a given drive attempting to become the synchronisation Producer at least one of these modules will have this bit clear. For a module attempting to become inter-module synchronisation Consumer it indicates that it has not requested to become Producer, and that a Producer has been located in another slot and is being used as the source of synchronisation data. (If a Producer has been located in another slot but the rate of the synchronisation data is not compatible with the Consumer then this bit will be cleared because the Producer - although located - is not being used as the source of synchronisation data.)
3	Synchronisation Producer Output within Specification	This bit is relevant only for modules which have been designated as the inter-option module synchronisation Producer and CTSync Slave. This bit indicates that the signal being provided by the inter-option module synchronisation Producer is within the specified tolerance of the drive, and that the drive is now locked to the CTSync Slave's frequency. Note that this bit may only be set if bits 2 to 0 of this parameter are 1,0,1, indicating that the module is Producer (0,1) and that the Producer role has been achieved (1).
7 to 4	<i>Reserved</i>	<i>Read as zero</i>

## 5.8 Menus 18,19 - Application Parameters

These two menus are designated as application parameters since they are all free to be used for whatever purpose the user wants.

Both menus are identical in their layout. All parameters are read/write access to the SM-Applications Lite (and via comms), but may be read-only on the drive's keypad.

#1x.01	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.

#1x.02- #1x.10	Integer, read-only		
Access	RW (RO drive)	Range	Signed 16-bit
Default	0	Update Rate	N/A

#1x.11- #1x.30	Integer, read-write		
Access	RW	Range	Signed 16-bit
Default	0	Update Rate	N/A

#1x.31- #1x.50	Bit, read-write		
Access	RO	Range	0/1
Default	0	Update Rate	N/A

Parameters #1x.11-#1x.50 are saveable in the drive's non-volatile memory.

## 5.9 Menu 20 - Application Menu

This menu, like menus 18 and 19, contains parameters that do not affect the operation of the drive and therefore can be used for general purpose.

**NOTE** This menu is NOT saved in the drive's non-volatile memory. Instead it can be stored in the SM-Applications Lite module's flash memory upon request. If more than one SM-Applications Lite module is fitted, only one should be configured to store and restore this menu for obvious reasons.

If parameter #81.21 is set, this menu will be saved and restored by the SM-Applications Lite module.

<b>#20.01- #20.20</b>	<b>Integer, read-write</b>		
<b>Access</b>	RW	<b>Range</b>	Signed 16-bit
<b>Default</b>	0	<b>Update Rate</b>	N/A

These parameters are standard integers giving them a range of -32768 to 32767.

<b>#20.21- #20.40</b>	<b>Long integer, read-write</b>		
<b>Access</b>	RW	<b>Range</b>	Signed 32-bit
<b>Default</b>	0	<b>Update Rate</b>	N/A

These parameters are long-integers, which give them a range of  $-2^{31}$  to  $2^{31}-1$ . On the drive display/keypad if the value exceeds the maximum that can be displayed (9,999,999), "-----" will be shown. It is not possible to enter values larger than the maximum display value on the keypad.

---

# 6 Communications

---

## 6.1 SM-Applications Lite mapping parameters

The SM-Applications Lite has internal parameters that can be written to or read from by fieldbus options also fitted to the Unidrive SP. This can provide a convenient way to communicate between 2 fieldbuses. These parameters are shown in the table below.

**Table 6.1 SM-Applications Lite module internal parameters**

SM-Applications Lite Parameters	Parameter Reference	Direct to Slot 1	Direct to Slot 2	Direct to Slot 3
_Pxx% PLC Registers	#70.xx	#100.xx	#130.xx	#160.xx
_Qxx% PLC Registers	#71.xx	#101.xx	#131.xx	#161.xx
_Rxx% PLC Registers	#72.xx	#102.xx	#132.xx	#162.xx
_Sxx% PLC Registers	#73.xx	#103.xx	#133.xx	#163.xx
_Txx% PLC Registers	#74.xx	#104.xx	#134.xx	#164.xx
_Uxx% PLC Registers	#75.xx	#105.xx	#135.xx	#165.xx
Local Configuration Parameters	#81.xx	#111.xx	#141.xx	#171.xx
Timer Function Parameters	#85.xx	#115.xx	#145.xx	#175.xx
Digital I/O Parameters	#86.xx	#116.xx	#146.xx	#176.xx
Status Parameters	#88.xx	#118.xx	#148.xx	#178.xx
General Parameters	#90.xx	#120.xx	#150.xx	#180.xx
Fast Access Parameters	#91.xx	#121.xx	#151.xx	#181.xx

The fieldbus interface module reads and writes data directly to and from the internal registers in SM-Applications Lite. The fieldbus interface module can read data from and write data to an SM-Applications Lite module fitted in any slot in the Unidrive SP, simply by specifying the target parameter as shown in Table 6.1.

**NOTE**

If a single SM-Applications Lite module is fitted to the Unidrive SP, normal SM-Applications Lite parameter references can be used without problem, as the SM-PROFIBUS-DP will automatically divert them to the SM-Applications Lite module.

### 6.1.1 Example configuration 1

Consider a Unidrive SP with the following configuration:

- Slot 1 - Vacant
- Slot 2 - SM-Applications Lite module
- Slot 3 - SM-PROFIBUS-DP module

If a parameter read request comes over the PROFIBUS-DP network to read #71.08, this will be re-directed to the SM-Applications Lite module in the lowest slot number, i.e. slot 2. The value in \_Q08% from slot 2 will be returned.

If a parameter read request comes over the PROFIBUS-DP network to read #131.08, this will be sent straight to the SM-Applications Lite module in slot 2. The value in \_Q08% from slot 2 will be returned.

If a parameter read request comes over the PROFIBUS-DP network to read #101.08, this will be sent straight to the SM-Applications Lite module in slot 1. As there is no SM-Applications Lite module fitted in slot 1, an error message will be returned, indicating that the parameter does not exist.

If, for example, an SM-DeviceNet module is fitted to slot 1, you could use the direct slot parameter reference to read or write data, giving a simple communications gateway between DeviceNet and Profibus-DP.

### 6.1.2 Example configuration 2

Consider a Unidrive SP with the following configuration:

- Slot 1 - SM-Applications Lite module
- Slot 2 - SM-Applications Lite module
- Slot 3 - SM-PROFIBUS-DP module

If a parameter read request comes over the PROFIBUS-DP network to read #71.08, this will be re-directed to the SM-Applications Lite module in the lowest slot number, i.e. slot 1. The value in `_Q08%` from slot 1 will be returned.

If a parameter read request comes over the PROFIBUS-DP network to read #131.08, this will be sent straight to the SM-Applications Lite module in slot 2. The value in `_Q08%` from slot 2 will be returned.

If a parameter read request comes over the PROFIBUS-DP network to read #101.08, this will be sent straight to the SM-Applications Lite module in slot 1. The value in `_Q08%` from slot 1 will be returned.

#### NOTE

If dual SM-Applications Lite modules are fitted to the Unidrive SP, it is best to access SM-Applications Lite module parameters using the direct slot parameter references. If normal SM-Applications Lite module parameters are used, and the SM-Applications Lite module is removed from slot 1, these parameter references will be re-directed to slot 2 instead.

### 6.1.3 Example configuration 3

Consider a Unidrive SP with the following configuration:

- Slot 1 - SM-Applications Lite module
- Slot 2 - SM-Applications Lite module
- Slot 3 - SM-Applications Lite module

The SM-Applications Lite module does not allow reading from or writing to Direct slot parameter references, so data transfer must be done another way.

If the SM-Applications Lite module in slot 1 wanted to read register #71.08 from the SM-Applications Lite module in slot 3, this must be done through one of the 32bit application parameters (#20.21-#20.40). The SM-Applications Lite module in slot 1 must write the register data to the application parameter. The SM-Applications Lite module in slot 3 can then read that parameter.

Code in slot 1 :

```
#20.21 = #71.08
```

Code in slot 3 :

```
#71.08 = #20.21
```

This data can also be accessed by the SM-Applications Lite module in slot 2.

#### NOTE

The SM-Applications Lite module cannot access the #1xx.xx internal menus. It must access them directly by the parameter reference number. e.g. if you wanted to read parameter #104.35 you would need to use #74.35.

---

# 7 DPL programming

---

This chapter covers:

- Basic DPL program structure and syntax
- Basic DPL commands
- New features offered by the SM-Applications Lite module



This is by no means a complete guide to the DPL language. 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. The SYPT Workbench creates this section for the user. It basically consists of:

- Program title
- Program author
- Program version number

### 7.1.1 Aliases

Immediately below the header the user may enter a section of *aliases*. Aliases are used to 'replace' various expressions or constants:

- a numerical constant expression
- the address of a register or parameter
- a DPL expression or statement

Aliases are created with the \$DEFINE statement.

```
$DEFINE name value
```

For example it is good practice to use aliases to give names to all drive parameters used in a program.

```
$DEFINE PRESET_REF_1 #1.21  
$DEFINE PRESET_REF_2 #1.22  
$DEFINE SPEED_FB #3.02
```

It is also recommended to have the alias name in UPPER-case letters in order to help distinguish them from normal variables.

#### NOTE

It is recommended that aliases representing integer values have a '%' symbol appended to the alias name. In graphical programming tools (QLD/FBD), SYPT will treat all aliases without a % symbol as floating-point values. Hence they will be rejected on LD or integer only inputs.

The \$DEFINE directive does NOT produce any code, nor does it speed up the execution time of your program - it simply allows you to refer to something with a different name.

## 7.2 Tasks

A DPL program is separated into 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	3	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. No other tasks can run until this task is completed.
BACKGROUND	1	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	2	Task executed on a fixed timebase (between 1-200ms) used for some time related operations, for example to generate a ramp profile. This task is now synchronised to the drive's level 2 control loop and can be used in place of the old Encoder task.
POS0 POS1	4	Two real-time tasks that run synchronously to a multiple of the drive control loops (range from 250us to 8ms). These tasks are commonly used to control the drive speed and/or current loop in applications such as positioning. The POS0 task is first to run, followed immediately after by the POS1 task.
EVENT	5	Event tasks only run when a certain event occurs. Events can be raised from various sources such as other option modules in the Unidrive SP or the user program. EVENT tasks are the highest priority so usually only have a very small number of instructions. They can be likened to interrupt service routines.
EVENT1	5	See above description.
EVENT2	5	See above description.
EVENT3	5	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.

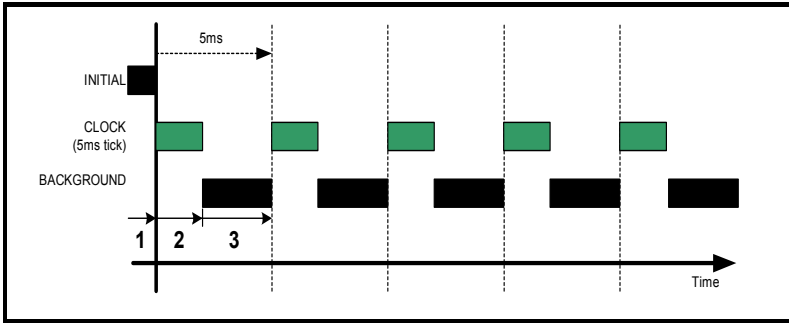
**NOTE**

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

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:

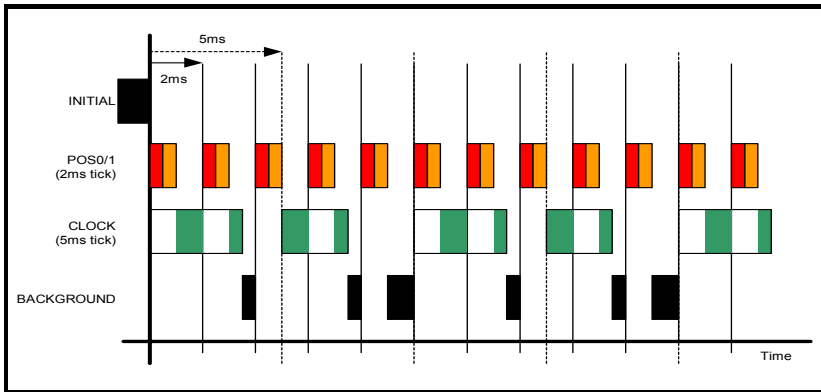


Key:

1. INITIAL task has exclusive control. No other tasks can run
2. CLOCK task which has a higher priority than the BACKGROUND task runs.
3. CLOCK task has finished and now the BACKGROUND task can run - 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 get a look-in, or a processor overload trip will occur.

The following diagram shows what happens when the POS tasks are used 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 #81.04 can be used to determine how heavily loaded the SM-Applications Lite module is.

## 7.2.1 EVENT tasks

There are four event tasks provided. The event tasks can be triggered on:

- 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 name speed% is NOT the same as SPEED%).
- SYPT QuickLD and FBD editors will only allow the use of variables no longer than 16 characters including any % sign.

### 7.3.3 Initialisation of variables

All variables must be given an initial value before they can be used. This is typically done within the INITIAL task. For example,

```
Initial {
  speed_sp% = 0
  ramp% = 0
}
```

### 7.3.4 Scope and lifetime of variables

Variables can either be global or local. All variables declared in DPL programs are global. i.e. they can be accessed and altered by any task, with the exception of variables within a User Defined Function Block which are local (i.e. cannot be accessed from outside the user defined function block).

No DPL variables survive a reset of the SM-Applications Lite module. Remember that resetting the drive from a tripped condition will also cause a reset.

### 7.3.5 Dynamic arrays

A DPL program may contain arrays of either integer or floating-point variables. Only 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. Eg:

```
DIM myarray%[20]    // Integer array having 20 elements
DIM array2[30]     // Floating point array having 30 elements
```

The elements in an array are numbered 0 to number\_of\_elements - 1. So from the above example, the first element of myarray%[] is:

```
myarray%[0]
```

and the last is:

```
myarray%[19]
```

Two functions are provided that can be used at run-time to determine the upper and lower bounds of an array. These are UPPER and LOWER. for myarray%[], UPPER will return 19 and LOWER will return 0.

### 7.3.6 Constant arrays

Constant arrays, as the name suggests, contain fixed pre-defined values. The values of the constant array are defined within the DPL program by using a special section (see CONST in the on-line help). Only integer values can be defined.

The advantage of constant arrays is that the size of the array is only limited by the amount of available program space - and not variable RAM. The program space is 384kb - it is used to store the compiled DPL file, constant array data, and optionally, the DPL file itself. RAM size is 80kb.

### 7.3.7 Storage space - number of variables

All variables, dynamic 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 period (.) followed by the bit number between 0 and 31.

Example 1 (simple variable):

```
flags% = 0    // initialise all 32 bits to 0
flags%.0 = 1 // set bit 0 to 1

// now test to see if bit 0 AND bit 1 are set to 1.
IF flags%.0 & flags%.1 = 1 THEN
    PRINT "Test satisfied."
ENDIF
```

Example 2 (arrays):

```
DIM myarray%[10]
...
IF myarray%.1[4] = 1 THEN;test bit 1 of element #4.
  PRINT "Test satisfied."
ENDIF
```

Note: The bit number must be a constant number - variables are not allowed.

### 7.3.9 PLC registers

The 'PLC' area is a special range of pre-defined 32-bit registers. The PLC registers are split into 6 sets of 100 parameters numbered 00 to 99. The registers can also be accessed from within a user DPL program by a special variable name or array name. Four of the register sets are also saveable in the SM-Applications Lite module *flash* memory.

See section 5.4 *Menus 70-75 - PLC Registers* for further information on PLC registers.

### 7.3.10 RAM files

RAM files enable the user to store 'files' in the user RAM of the SM-Applications Lite module. 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 online help.

## 7.4 Parameters

Parameters are categorised into two sets:

- Drive Parameters
- SM-Applications Lite Parameters

Drive parameters are ones which reside in the host drive. The majority of them affect the operation of the drive, however a few are set aside as "application parameters". These are menus 18, 19 and 20.

The SM-Applications Lite parameters are local and accessible only to the SM-Applications Lite module. These parameters provide access to additional features of the SM-Applications Lite module, and give faster access to some drive parameters.

#### NOTE

The SM-Applications Lite module always guarantees that the drive parameter database it uses matches that of the host Unidrive SP. When a SM-Applications Lite module is fitted to a Unidrive SP for the first time and powered up the word "Loading" may appear on the drive display for a few seconds. This indicates the SM-Applications Lite module is synchronising databases. This will only occur the first time the module is fitted to the drive. Subsequent power-ups will shown "Loading" only for a very short time.

### 7.4.1 Reading and writing parameters

Reading and writing parameters is achieved by using the # command. Parameters are accessed in a menu.parameter format in the same way as they are accessed on the drive's keypad.

For example, to read the speed feedback parameter (parameter 03.02), use:

```
speed% = #3.02
```

to write to a speed reference parameter (eg. 01.22), use:

```
#01.22 = 1500
```

Note that the leading zero in the menu/parameter field is optional. For example #3.02, #03.02, #03.2 and #3.2 will access exactly the same parameter.

### 7.4.2 Floating point parameters

Dealing with floating point parameter 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. This command automatically does a conversion from float to int and visa-versa.

For example, parameter #1.19 has a range of 0.000 - 0.099. Reading the parameter using:

```
speed_fine% = #INT1.19
```

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

```
#INT1.19 = 45
```

will set the parameter to 0.045 (same as #1.19=0.045). The benefit of this is that the DPL program can use integer variables (%) instead of floating-point, thus providing a speed advantage.

## 7.5 Operators

DPL offers all the standard operators as follows:

**Table 7.3 Standard operators in precedence order**

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

**Table 7.4 Conditional operators in precedence order**

Operator	Meaning
=	Equality
<	Less than
>	Greater than
<=	Less then or equals
>=	Greater than or equals
<>	Inequality
AND	Boolean AND
OR	Boolean OR
NOT	Boolean NOT

## 7.6 Basic DPL commands

The DPL language implemented for the SM-Applications Lite module is backwardly compatible with the UD70 product for UMV 4301, 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 SM-Applications Lite module

These are new commands that have been introduced into the DPL language for the SM-Applications Lite module and were not present in the UD70 for UMV 4301.

#### FOR loop

This is new to the SM-Applications Lite module.

```
FOR variable = integer_expression to integer_expression [STEP constant]
  statements
LOOP
```

The STEP part is optional.

#### 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

The CASE statements operate in the same way as programs like Visual Basic in that the program flow will NOT drop through to the next CASE as it does in the C programming language.

## MAX\_INT, MIN\_INT, MIN\_FLOAT, MAX\_FLOAT

These are special predefined keywords that are recognised by the DPL compiler and replaced by the appropriate numeric value.

**Table 7.5 Min/Max**

Keyword	Value
MIN_INT	-2147483648
MAX_INT	2147483647
MIN_FLOAT	-3.40282e+038 (single precision model) -1.79769e+308 (double precision model)
MAX_FLOAT	3.40282e+038 (single precision model) 1.79769e+308 (double precision model)

## UPPER/LOWER

These functions will take an array as a parameter and will return the upper and lower array index respectively. For example:

```
// Create an array of 1000 elements
DIM array%(1000)

// now,

l% = LOWER(array%) // will return the value 0
u% = UPPER(array%) // will return the value 999.

// get the sum of all values in array%
total%=0
FOR i% = LOWER(array%) TO UPPER(array%)
    total% = total% + array%(i%) //add array element value to total
LOOP
```

## TRUNC

This is used to convert a floating point value to integer, truncating rather than rounding. For example:

```
// Initialise floating point variable
floatval = 1.56

int1% = floatval // auto-cast rounds to 2.
int2% = INT(floatval) // explicit cast with INT rounds to 2
int3% = TRUNC(floatval) // explicit cast with TRUNC gives 1
```

## 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 #90.12-#90.15.*

```
BACKGROUND {
... some code
// Schedule local event1 task with reason code of 45.
a% = SCHEDULEEVENT(0, 1, 45)
... some more code
}

EVENT1 {
IF #90.13 = 45 THEN
    // task scheduled from DPL
ENDIF
}
```

## GETPARATTR

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

```
(max%, min%, flags%) = GETPARATTR(menu%, par%)
```

## CModeXfer

This allows the user to change the drive type without any fitted option modules getting a hard reset. It allows for a smoother drive type change. While the drive type 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.

## 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 parameter #81.49.

```
SETUSERID(101) // set #81.49 to 101.
```

**AssRAM**  
**UnassRAM**  
**RamLength**  
**SetRamLength**

These commands allow the programmer to use the RAM files within the SM-Applications Lite module. 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.

**OpenReadSmartCard**  
**OpenWriteSmartCard**  
**CloseSmartCard**  
**ReadSmartCardByte**  
**WriteSmartCardByte**  
**GetNextSmartCardFile**  
**ReadReadOnlyBit**  
**WriteReadOnlyBit**

These commands allow the user to utilise the Unidrive SP's SMARTCARD. For further information contact LEROY-SOMER.

### **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. This list is too large to go into in this User Guide, so please refer to the on-line help.

## **7.7 User defined function blocks**

### **7.7.1 Overview**

SYPT 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 SYPT 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 through 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 SM-Applications Lite 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 SYPT 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 SYPT 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 SYPT 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.

#### NOTE

Due to the fact that the SYPT Workbench was developed for the UD70, the tools used to create UDFB's still enforce the restriction on input and output arguments and the order of floating point and integer variables. Therefore the full flexibility of the SM-Applications Lite module's implementation cannot be realised with SYPT Workbench V1.

More information on the restrictions can be found in the on-line help.

### 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 are not available in the initial section..

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 QuickLD diagrams.

Below is an example of a simple UDFB that adds two numbers and scales by a pre-defined amount (0.5):

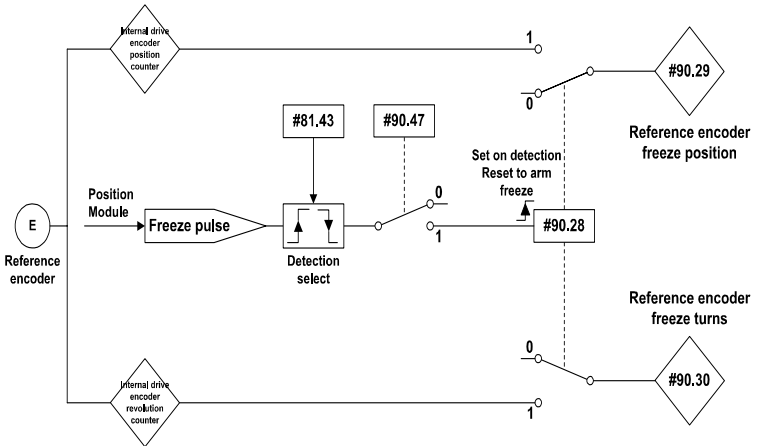
```
(output%) = _simplefb(input1%, input2%) {  
  // initialisation code:  
  
  scale% = 500    // initialise a local variable  
  
  FBbody  
  // main body code:  
  
  output% = input1% + input2% * scale% / 1000  
  
}
```

# 8 SM-Applications Lite Freeze and Marker

## 8.1 Freeze input

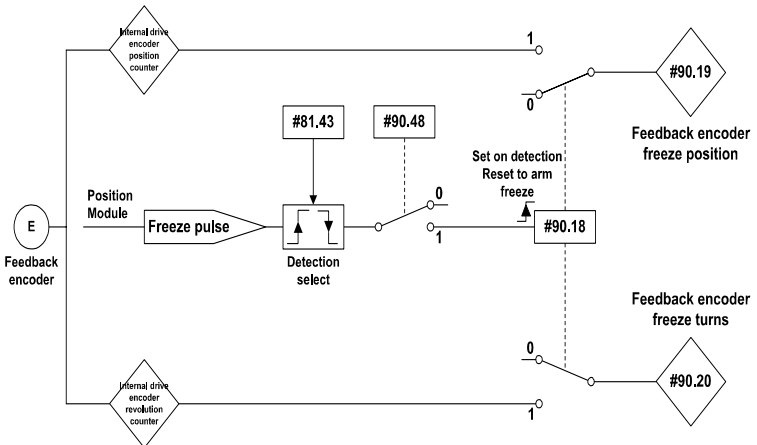
The SM-Applications Lite is unlike the SM-Applications module in that it does not have digital I/O which can be used to 'freeze' the reference and feedback encoder counters. However, another module in the same drive can be used to freeze the encoder data in the SM-Applications Lite. For instance, setting parameter x.40 for an SM-Universal Encoder Plus module to 1 allows other slots in the same drive to freeze data.

**Figure 8-1 SM-Applications Lite Reference Freeze Input**



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

**Figure 8-2 SM-Applications Lite Feedback Freeze Input**



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

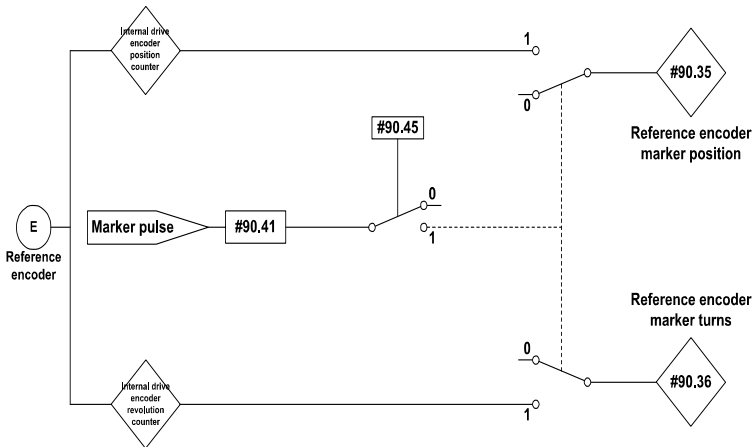
The freeze position of both the reference and feedback encoders can be captured on the rising or the falling edge of the freeze pulse. This is determined by setting parameter #81.43 to either 0 (rising edge) or 1 (falling edge). Parameter #81.42 enables the position to be written to parameters #90.19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

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

## 8.2 Marker pulse

The SM-Applications Lite is able to cache the position and rev count at the point when a Marker pulse is seen on the reference or feedback encoders.

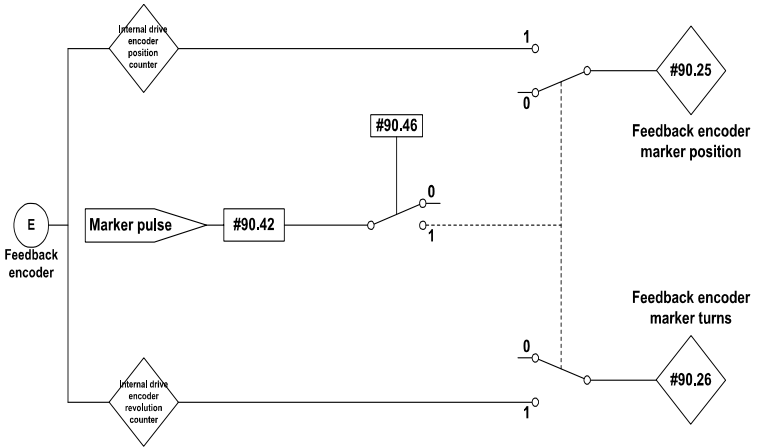
**Figure 8-3 SM-Applications Lite Reference Marker**



The marker position is cached into parameter #90.35 and the marker revolution counter is cached into parameter #90.36.

The user sets #90.41 to a zero and the drive sets #90.41 to a 1 EVERY time a marker is detected. The user must ensure that the marker data is consumed before the next marker pulse.

**Figure 8-4 SM-Applications Lite Feedback Marker**



The marker position is cached into parameter #90.25 and the marker revolution counter is cached into parameter #90.26.

The user sets #90.42 to a zero and the drive sets #90.42 to a 1 EVERY time a marker is detected. The user must ensure that the marker data is consumed before the next marker pulse.

---

# 9 Inter-Option Module Synchronisation

---

## 9.1 Overview

The Inter-Option Module Synchronisation scheme provides a mechanism to allow position control tasks on one or more modules to be synchronised to position control tasks on another module on the same Unidrive SP without the need for external wiring. Synchronisation is achieved by one module producing a trigger signal and one or more modules consuming the trigger signal. These modules are referred to as Producers and Consumers. The trigger signal is used to alter the scheduling of the POS engine.

If the module is a Producer the trigger may be provided at a rate of 250µs, 500µs, 1ms, 2ms, 4ms or 8ms i.e. Producer modules generate the trigger every POS period, as specified in parameter #81.12 .

The SM-Applications Lite module, when a Consumer, can run its Position Control tasks at the same rate or quicker than the Synchronisation Signal (trigger) from the Producer. If an SM-Applications Lite module is a Consumer and its position control task is set to run slower than that of the Synchronisation Signal (trigger) from the Producer, then it will be impossible for it to know the phase of the Producer, so it will not attempt to synchronise to the Producer's signal.

## 9.2 Inter-Option Synchronisation example

This example shows a scenario where the POS0 tasks of the modules in a Unidrive SP need to be synchronised.

The following table shows the settings for each of the 3 modules:

**Table 9.1 Parameter Settings**

	Parameter	Value	Description
Slot 1	#81.12	2	500µs position control task period
	#91.21	2	Inter-Option Sync Consumer
	#91.22	6	<i>Inter-Option Sync Consumer status achieved</i>
Slot 2	#81.12	1	250µs position control task period
	#91.21	2	Inter-Option Sync Consumer
	#91.22	6	<i>Inter-Option Sync Consumer status achieved</i>
Slot 3	#81.12	3	1ms position control task period
	#91.21	1	Inter-Option Sync Producer
	#91.22	5	<i>Inter-Option Sync Producer status achieved</i>

The italicised parameter #91.22 is a status parameter and does not require setting. It shows whether the module has achieved the Inter-option functionality specified in parameter #91.21. For more information refer to parameter #91.22 .

Before setting the parameters above, the POS tasks may be executing out of phase with each other as shown in Figure 9-1 *POS Task Execution BEFORE Inter-option Synchronisation* . After setting the parameters to the values shown above the POS tasks will be executing in phase with each other as shown in Figure 9-2 *POS Task Execution AFTER Inter-option Synchronisation* .

Figure 9-1 POS Task Execution BEFORE Inter-option Synchronisation

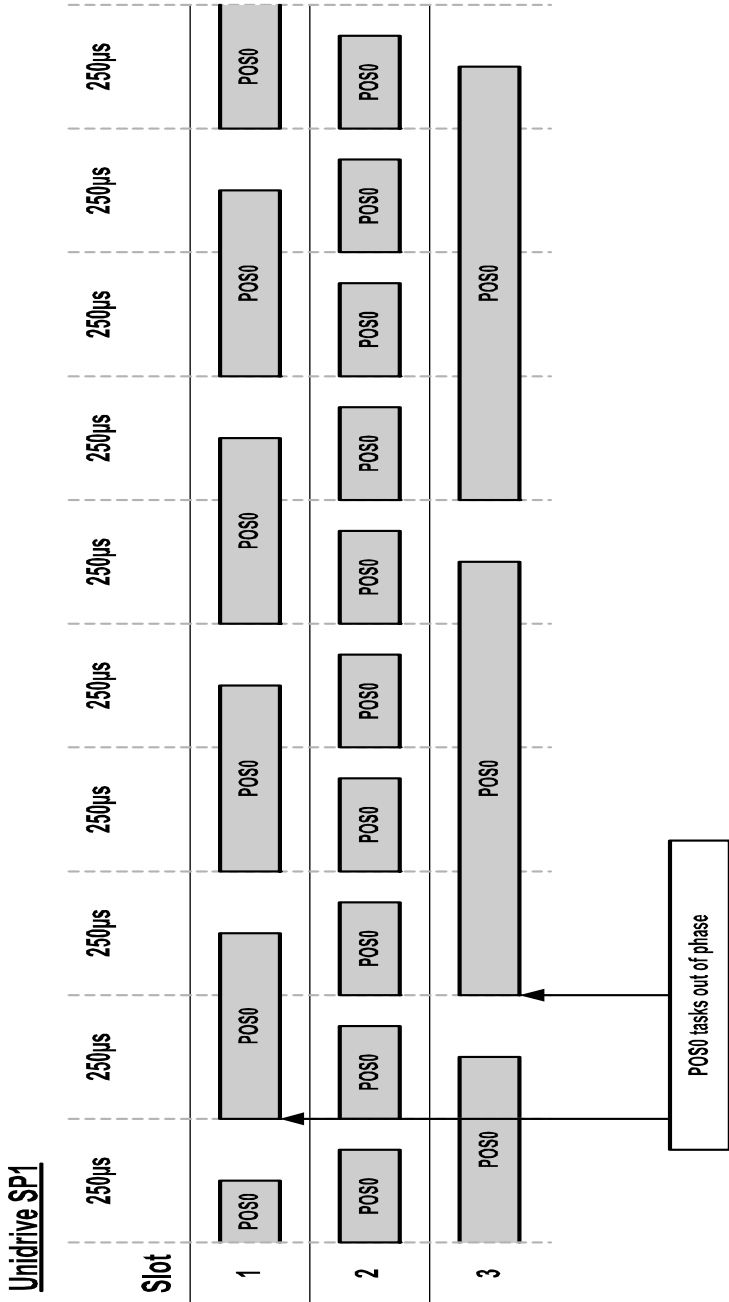
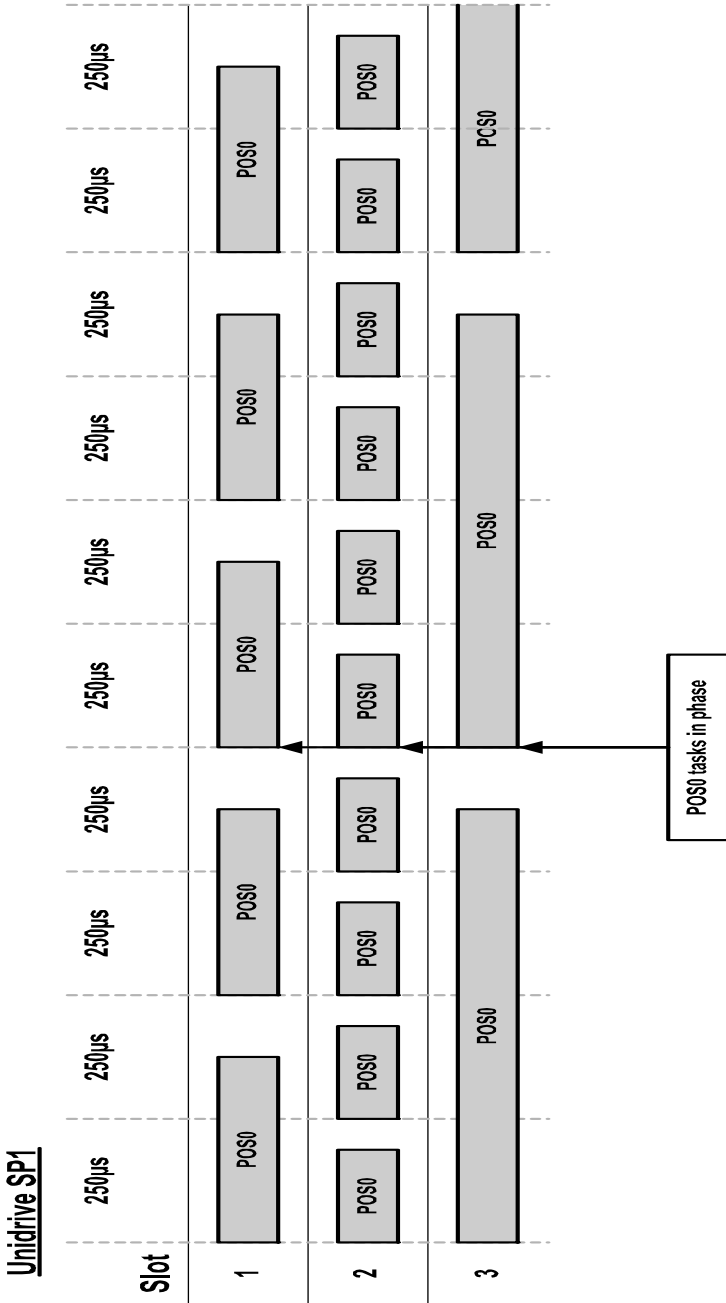


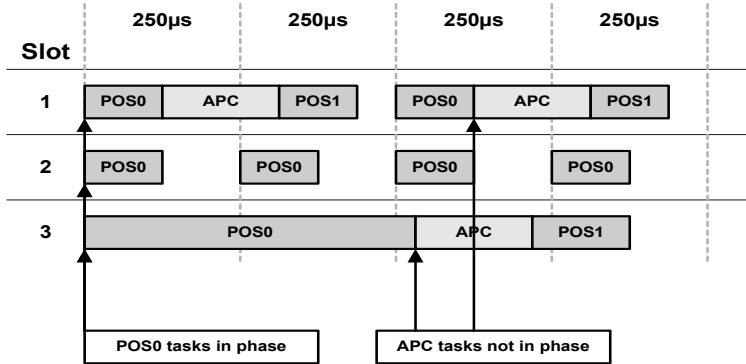
Figure 9-2 POS Task Execution AFTER Inter-option Synchronisation



### 9.3 Position control tasks

The position control tasks consist of POS0, APC and POS1. The tasks are run in this order (POS0 runs first followed by the APC which is followed by POS1). Because the POS0 and APC tasks may not take the same time to execute on different modules the APC and POS1 tasks cannot be phased. Therefore when using Inter-option synchronisation the user must be aware that the position tasks as a whole for each module will start in phase, not the individual tasks. Figure 9-3 *Position control tasks* shows an example of this.

Figure 9-3 Position control tasks



---

# 10 Diagnostics

---

This chapter details the following:

- Run-time errors and trip codes
- Handling of run-time errors
- Support

## 10.1 Run-time Errors

A run-time error is an error which occurs in a specific operation of the SM-Applications Lite module. 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) or a system error such as processor overload or *watchdog* time-out.

The action taken when an error occurs may be one or more of the following:

- User program may be halted or prevented from starting
- Drive may be tripped on **SLx.Er** (where x is the slot number) with the run-time error code placed into #MM.50
- Drive may be tripped on another **SLx.\*\*\*** code.
- The DPL **ERROR** task may be executed (if it exists).

Which of these occurs depends upon the type of error and the setting of the global run-time trip enable parameter #MM.14. This is detailed in section 10.3, below.

## 10.2 Unidrive SP Trip Display Codes

The table below shows the possible trip codes that will be displayed on the Unidrive SP when an error is detected in the SM-Applications Lite module which instigates a drive trip. Remember, not all run-time errors instigate a drive trip.

**Table 10.1 Unidrive SP Trip Display Codes**

Slot Where Trip Was Initiated			Fault	Description
Slot 1	Slot 2	Slot 3		
SL1.HF	SL2.HF	SL3.HF	Hardware Fault	Unidrive SP has detected that an option module is present, but is unable to communicate with it.
SL1.tO	SL2.tO	SL3.tO	Watchdog Timeout	Indicates a user program which has utilised the <i>watchdog</i> feature has failed to issue the WDOG command within 200ms.
SL1.Er	SL2.Er	SL3.Er	Error	Run-time trip generated by the SM-Applications Lite module either due to a user DPL program error or some other event. The actual error code is placed into #MM.50.
SL1.nF	SL2.nF	SL3.nF	Not Fitted	SM-Applications Lite module was disconnected while operational, or SM-Applications Lite module has crashed. This trip will also occur if a Unidrive SP slot is configured for a SM-Applications Lite module, but the module is not fitted in the slot.
SL1.dF	SL2.dF	SL3.dF	Different Fitted	This trip will occur when an SM-Applications Lite module is fitted to a slot previously occupied by another solutions module, or is fitted to a previously unused slot.

## 10.3 SM-Applications Lite Run-time Error Codes

If the SM-Applications Lite module detects an error during operation the error code will be placed in the following parameter:

#81.50	SM-Applications Lite Error Code		
Access	RO	Range	0 to 255
Default	N/A	Update Rate	On error

For certain errors the user may select if the Unidrive SP should trip as well. This is configured with the Global Run-time Trip enable parameter:

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

If set to 1 (On), the drive will trip on ALL run-time errors.

The table below shows the error codes and their meaning as well as if the drive will trip, the User program will stop and whether the DPL ERROR task will run.

### Notes:

- **“May”** under Drive Trip indicates that the drive will only trip if the global run-time trip enable parameter is set
- **“Not Run”** under Program Halted indicates that the error occurs at initialisation and the program will not be started.

**Table 10.2 SM-Applications Lite Error Codes**

Error Code	Reason	Trip Drive?	ERROR task?	Prog Halted?
39	User program stack overflow	Yes	No	Yes
40	Unknown error - please contact supplier	Yes	No	Yes
41	Parameter does not exist. User has attempted to read or write a non-existent parameter in the DPL program.	May	Yes	Yes
42	Attempt to write to a read-only parameter.	May	Yes	Yes
43	Attempt to read from a write-only parameter.	May	Yes	Yes
44	Parameter value out of range. (User has written an illegal value to a parameter within a DPL program.) If parameter #MM.17=0 the value written will be automatically limited and no error will occur.	May	Yes	Yes
45	Invalid synchronisation modes	Yes	No	Not Run
50	Maths error - divide by zero or overflow.	May	Yes	Yes
51	Array index out of range. E.g. arr%[20] where arr% has only been DIMensioned to 19 elements.	May	Yes	Yes
52	Control word user trip. Instigated by setting the trip bit in the control word #90.11	Yes	No	No
53	DPL program incompatible with target. For example, downloading a program compiled for UD70.	Yes	N/A	N/A
54	DPL task overrun. This occurs if the DPL code within a real-time task (e.g. POS0) cannot be completed in time. Use parameter #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.	May	Yes	Yes
57	Function block does not exist.	Yes	Yes	Not Run

**Table 10.2 SM-Applications Lite Error Codes (Continued)**

Error Code	Reason	Trip Drive?	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 application module as Sync master	Yes	Yes	Yes
65	Invalid function block parameter(s). You have called a FB within a DPL program but one or more of the inputs are invalid.	Yes	Yes	Yes
66	User heap too large. The program has been compiled for a target that has more RAM than this one has. Occurs at startup.	Yes	No	Not Run
67	RAM file does not exist or a non-RAM file id has been specified.	Yes	Yes	Yes
68	The RAM file specified is not associated to an array.	Yes	Yes	Yes
69	Failed to update drive parameter database cache in Flash memory.	Yes	No	Not Run
70	User program downloaded while drive enabled. Will occur if #MM.37 = 1 and a program is downloaded.	May	No	Yes
71	Failed to change drive mode	Yes	No	Yes
73	Fast parameter initialisation failure	Yes	No	No
74	Over-temperature	Yes	Yes	Yes
75	Hardware unavailable. The user program attempted to access unavailable hardware. e.g. if access is made to digital I/O, RS485 port or CTNet on SM-Applications Lite module.	Yes	Yes	Yes
76	Module type cannot be resolved. Module is not recognised.	Yes	No	Not Run
77	Inter-option module comms error with module in slot 1.	Yes	Yes	Yes
78	Inter-option module comms error with module in slot 2.	Yes	Yes	Yes
79	Inter-option module comms error with module in slot 3.	Yes	Yes	Yes
80	Inter-option module comms error with module unknown slot.	Yes	Yes	Yes
81	APC internal error. See parameter #81.38 (e.g. use of uninitialised CAM function).	May	Yes	Yes
82	Communications to drive faulty.	May	Yes	Yes

## 10.4 Handling Run-Time Errors with the ERROR task

Certain run-time errors will cause the DPL ERROR task to be invoked if it exists. This provides a convenient way to safely handle the error condition and take whatever action is necessary, such as a controlled stop of the system or signalling of an alarm.

When an ERROR task runs, all other DPL tasks will have been stopped. Therefore the ERROR task has exclusive execution rights. Once the ERROR task has completed, the DPL program ends and no other DPL tasks operate (though it is possible to reset and restart the program - more details on this below).

**NOTE** Drive trips do not cause the ERROR task to run. Only certain DPL program errors do.

Within the ERROR task all standard DPL commands may be used as well as most function blocks. All drive and SM-Applications Lite parameters can be accessed.

The run-error code can be determined using this parameter:

#88.01	Error Status / Reset		
<b>Access</b>	RW	<b>Range</b>	0 to 9999
<b>Default</b>	N/A	<b>Update Rate</b>	On error

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

When the parameter is written to with a value of 1070 the SM-Applications Lite module will initiate a warm-restart of the drive and any other options. This can be used to restart the user program (providing auto-run #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 parameter #88.01 will result in any drive trip being automatically cleared as well as resetting all fitted options in the Unidrive SP. This behaviour is different to the UD70 product on UMW 4301 where the drive was not reset.

The task that caused a run-time error can be determined by reading parameter #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 parameter #10.38.

## 10.5 Support

The information from the parameters described below should always be noted before contacting your supplier for technical support.

### 10.5.1 Module Firmware

#81.02	Firmware - Major Version		
<b>Access</b>	RO	<b>Range</b>	00.00 to 99.99
<b>Default</b>	N/A	<b>Update Rate</b>	N/A

#81.51	Firmware - Minor Version		
<b>Access</b>	RO	<b>Range</b>	0 to 99
<b>Default</b>	N/A	<b>Update Rate</b>	N/A

The full version of the SM-Applications Lite module firmware version can be read for the corresponding slot. This manual was written for SM-Applications Lite modules fitted with V01.03.03 firmware. The table below shows how to construct the full firmware version from these values.

**Table 10.3 Firmware Version**

Major Version	Minor Version	Firmware Version
1.01	5	V01.01.05

---

# 11 Migration Guide

---

This chapter outlines some of the major differences between the UD70 product for UMV 4301 and the SM-Applications Lite module for Unidrive SP that may be helpful in converting (porting) user DPL programs.

## 11.1 Drive Parameter Differences

The Unidrive SP parameter set has quite a few differences compared to UMV 4301. All parameters accessed in a DPL program should ideally have been configured as #define's at the top of the program, thereby making conversions easier.

So in porting, you will have to go through the program by hand and locate all drive parameter references. The first step is to search for the # symbol, and then go through the program looking for any function blocks that may take parameter pointer inputs (e.g. PFixRead).

When parameters are located, make sure they are the same in the Unidrive SP and ensure that they are in the same units. Adjust the parameter number or scaling as appropriate.

## 11.2 UD70 Parameters

### 11.2.1 Setup Parameters

The set-up parameters can now be located in menu 15,16 or 17 whereas with UD70 they were always located in menu 17. It is strongly advised that new user DPL programs that change setup parameters are altered to use menu 81 which will eliminate the need to determine which of menu 15/16/17 is being used.

The table below outlines some of the major differences between the two setup menus:

**Table 11.1 Setup Parameter Changes**

Parameter	Description	Changes
#17.03	Line number of error	This is now #81.48.
#17.06	RS485 mode	Not supported. Parameter now unused.
#17.08	RS485 pointer	Not supported. Parameter now unused.
#17.09	RS485 pointer 2	Not supported. Parameter now unused.
#17.10	RS485 scale factor	Not supported. Parameter now unused.
#17.11	Clock tick time	With drive version < 01.05.00 the default is zero. With drive versions >=01.05.00 the default is 10ms.
#17.12	Position controller	This parameter now controls the scheduling rate for the new POS0/1 which are different to those of UD70 Speed and Encoder tasks.
#17.15	RS485 trip mode	Not supported. Parameter is now used to disable a module reset occurring when a drive trip is cleared.
#17.16	IO Link sync. source	IO link is not supported. Parameter is now used to select the encoder update rate.
#17.20	Flash power-down store	This parameter now takes immediate effect.
#17.21	Disable MD29MON	This parameter now controls the saving of menu 20.
#17.22	RS232 drive-drive	Not supported. Parameter now unused.
#17.23	EVENT trigger control	Not supported. Parameter now unused.
#17.24	ANSI 2-wire turn-around	Not supported. Parameter now unused.
#17.25+	Unused	There are now 51 set-up parameters.

## 11.2.2 Menus 18/19

These menus remain unchanged.

## 11.2.3 Menu 20

- Menu 20 now consists of 40 parameters rather than 50. Parameters #20.21 to #20.40 are now full signed 32-bit parameters.
- Previously the first 20 parameters of menu 20 were considered to be reserved for communications options. This is now no longer the case since communications parameters are set up in menu 15/16/17, therefore all of menu 20 is free for use.
- The saving of menu 20 has also changed. In order to save the parameters you must set parameter #81.21 to a 1 and then set #81.19 or #81.20. The reason for this change is that you can now potentially have two or three SM-Applications Lite modules fitted into the drive and since menu 20 is saved and restored from the flash memory of the module, only one module can be configured to look after menu 20 otherwise it will be incorrectly restored.

## 11.2.4 PLC Registers

- Good news - there are now an extra 200 **saveable** registers available. These are in register banks T and U (or menus 74 and 75). The P/Q/R/S register sets remain as they were, however since the built-in position controller does not use the Q registers, they are completely free for use.

## 11.2.5 Menu 90 Parameters

Substantial changes have occurred to parts of this menu and programs WILL have to be modified to take them into account. The encoder position parameters #90.01/#90.03 now provide the full position information (including fine position) scaled such that the full 32-bit value represents one revolution. Separate revolution counters are available in #90.02 and #90.04.

# 11.3 General Features

## 11.3.1 Hardware

- RS232 programming port is no longer available. Programming is now done via the CT-RTU protocol through the RJ45 serial port on the front of the drive. (The old DPL Toolkit software is not supported.)

## 11.3.2 DPL Language

DPL remains backwardly compatible. The following enhancements have been made though:

- Addition of new constructs such as FOR...LOOP and SELECT...CASE.
- Nesting: The stack size is now allocated on a per-task basis and is larger than that of the UD70, so more nesting is allowed.
- New data type of single-precision floating point has been added that offers a small execution time advantage over double-precision (which remain the default). Single or double precision is specified as a global program option using the new \$FLT SINGLE directive in the program header.

**NOTE** The single-precision data type is not a direct replacement for double-precision and users should make sure that the single precision type offers sufficient precision for their application.

### 11.3.3 Maths

- Expression complexity is now greatly improved in that on the UD70 you were limited to how much you could do in one expression. Now you can create expressions with much greater complexity.
- A new casting operator TRUNC has been added. This provides a float to integer conversion, truncating rather than rounding the result.
- Floating point comparison: With the UD70 a rather imprecise and unpredictable fuzzy compare was used for comparing floating point values. Now the SM-Applications Lite module uses a standard comparison method, as used in other programming languages, that is not fuzzy. However this may lead to an unexpected side effect. For example,

```
f = 1.2 * 3
IF f = 3.6 THEN
    // This would not yield true.
ENDIF
```

If checking for equality or not equality, use a range. For example:

```
f = 1.2 * 3
IF f > 3.59 or f < 3.61 THEN
    // This would yield true.
ENDIF
```

### 11.3.4 Tasks

The tasks have been changed.

- The ENCODER and SPEED tasks are no longer used. Instead the POS0 and POS1 tasks have been added. Also, the CLOCK task can be used as a replacement for the ENCODER task. The POS0, POS1 and CLOCK tasks run synchronously to the drive (like the SPEED task did), but you can now specify a multiple of that time - from 250µs to 8ms for the POS0 and POS1 and 5ms to 200ms for the CLOCK task. Users may still use the names SPEED and ENCODER since they are aliases to POS0 and POS1 (when using SYPT Workbench V1 this may be beneficial since there is an issue with breakpoints with POS0/1), but be aware that the timing **is** different and code may have to be altered to take this into account. Note also that the timing does not alter depending on drive switching frequency, as it did on UD70.
- Three new EVENT tasks have been added, but this should not cause any problems migrating programs.
- EVENT tasks can now be scheduled from DPL using the new SCHEDULEEVENT function block.

### 11.3.5 Position Controller

The SM-Applications Lite module has a built in Advanced Position Controller, however this is completely different to the UD70 Advanced Position Controller. For information on this refer to the Advanced Position Controller User Guide.

### 11.3.6 User Defined Function Blocks

- The UD70 had a limit of 10 integer inputs and 10 integer output. There were also alignment restrictions. The SM-Applications Lite module has none of the restrictions. Number of inputs and outputs are limited only by memory. There should be no migration problems. (Note: The SYPT Workbench currently still imposes the old limitations on programs developed for the SM-Applications Lite module).

## 11.4 SM-Applications Porting Aid

### 11.4.1 Overview

The SYPT Workbench v1.6.0+ now includes a Porting Aid which will warn users of any features that are different on the SM-Applications Lite compared with the UD7X modules. The warnings will be displayed by SYPT Workbench in an error window. This error window will include the line number that the error/porting information is related to, so by double-clicking the line in the error window SYPT Workbench will show the DPL source line containing the changed item. This porting aid can be activated by using the new \$PORTING directive after the main header information:

```
$AUTHOR Kevin Vedmore
$COMPANY LEROY-SOMER
$TITLE Test Program
$VERSION v1.0
$DRIVE UNIDRVSP
$PORTING
```

### 11.4.2 Reported Differences

The following kinds of information will be reported by SYPT Workbench when the \$PORTING directive is used:

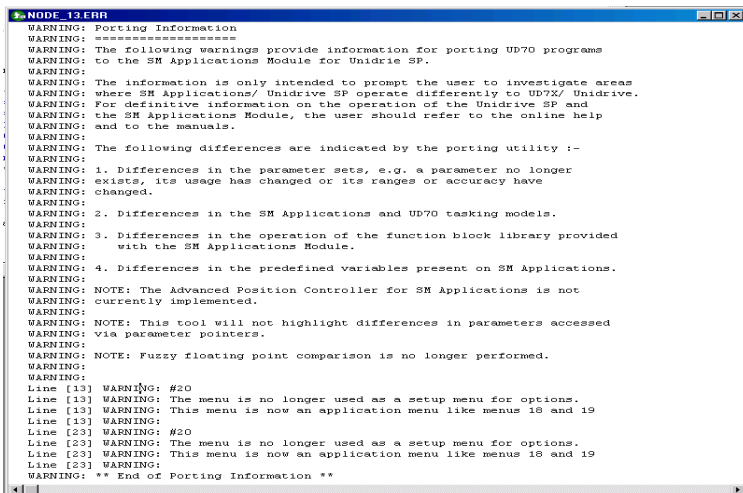
1. POS0 and POS1 tasks will be recommended instead of the older Speed, Torque and Encoder tasks. The Clock task will be recommended as an alternative to using a Pos task in place of an Encoder task.
2. Differences in the Unidrive SP or SM-Applications Lite parameter sets, e.g. the number of decimal places has changed, names and usage has changed and the parameter no longer exists or has moved.

Note: Specific warnings about the number of decimal places changing will only be shown if a parameter is accessed using the #INT notation. An overall warning about decimal place changes will be shown for non-#INT parameter accesses.

3. Different operation of RTL and OS function blocks.

An example of the type of messages given by the Porting Aid is shown below.

Figure 11-1 Porting window information



```
% NODE 13 ERR
WARNING: Porting Information
WARNING: *****
WARNING: The following warnings provide information for porting UD70 programs
WARNING: to the SM Applications Module for Unidrive SP.
WARNING:
WARNING: The information is only intended to prompt the user to investigate areas
WARNING: where SM Applications/ Unidrive SP operate differently to UD7X/ Unidrive.
WARNING: For definitive information on the operation of the Unidrive SP and
WARNING: the SM Applications Module, the user should refer to the online help
WARNING: and to the manuals.
WARNING:
WARNING: The following differences are indicated by the porting utility :-
WARNING:
WARNING: 1. Differences in the parameter sets, e.g. a parameter no longer
WARNING: exists, its usage has changed or its ranges or accuracy have
WARNING: changed.
WARNING:
WARNING: 2. Differences in the SM Applications and UD70 tasking models.
WARNING:
WARNING: 3. Differences in the operation of the function block library provided
WARNING: with the SM Applications Module.
WARNING:
WARNING: 4. Differences in the predefined variables present on SM Applications.
WARNING:
WARNING: NOTE: The Advanced Position Controller for SM Applications is not
WARNING: currently implemented.
WARNING:
WARNING: NOTE: This tool will not highlight differences in parameters accessed
WARNING: via parameter pointers.
WARNING:
WARNING: NOTE: Fuzzy floating point comparison is no longer performed.
WARNING:
WARNING:
Line [13] WARNING: #Z0
Line [13] WARNING: The menu is no longer used as a setup menu for options.
Line [13] WARNING: This menu is now an application menu like menus 10 and 19
Line [13] WARNING:
Line [23] WARNING: #Z0
Line [23] WARNING: The menu is no longer used as a setup menu for options.
Line [23] WARNING: This menu is now an application menu like menus 10 and 19
Line [23] WARNING:
WARNING: ** End of Porting Information **
```

# 12 Quick Reference

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

**Table 12.1 Parameters**

Parameter	Description	Range	Default
#81.01	Module option code	0-499	N/A
#81.02	Module firmware version	0-99.99	N/A
#81.03	DPL program status	0-3	N/A
#81.04	Available system resource %	0-100%	N/A
#81.11	Clock tick time (ms)	0-200ms	10
#81.12	POS task schedule rate	0-6	0
#81.13	Enable autorun	0-1	1
#81.14	Global run-time trip enable	0-1	0
#81.15	Disable reset on trip cleared	0-1	0
#81.16	Encoder Data Update Rate	0-3	0
#81.17	Enable parameter over-range trips	0-1	0
#81.18	Watchdog enable	0-1	0
#81.19	Save request	0-1	0
#81.20	Save on 'UU' trip	0-1	0
#81.21	Include menu 20 for save/restore	0-1	0
#81.37	Reject download if drive enabled	0-1	0
#81.38	APC Run-time trip	0-1	0
#81.39	Inter-option module Drive Synchronisation Status	0-3	0
#81.43	Freeze invert	0-1	0
#81.44	Task priority level	0-255	0
#81.48	DPL line number of error	32bit	0
#81.49	User program ID	16bit	0
#81.50	Run-time error code	0-255	0
#81.51	Minor software revision	0-99	N/A

**Table 12.2 SM-Applications Lite Error Codes**

Error Code	Reason	Trip Drive?	ERROR task?	Prog Halted?
39	User program stack overflow	Yes	No	Yes
40	Unknown error - please contact supplier	Yes	No	Yes
41	Parameter does not exist. User has attempted to read or write a non-existent parameter in the DPL program.	May	Yes	Yes
42	Attempt to write to a read-only parameter.	May	Yes	Yes
43	Attempt to read from a write-only parameter.	May	Yes	Yes
44	Parameter value out of range. (User has written an illegal value to a parameter within a DPL program.) If parameter #MM.17=0 the value written will be automatically limited and no error will occur.	May	Yes	Yes
45	Invalid synchronisation modes	Yes	No	Not Run
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

**Table 12.2 SM-Applications Lite Error Codes (Continued)**

Error Code	Reason	Trip Drive?	ERROR task?	Prog Halted?
52	Control word user trip. Instigated by setting the trip bit in the control word #90.11	Yes	No	No
53	DPL program incompatible with target. For example, downloading a program compiled for UD70.	Yes	N/A	N/A
54	DPL task overrun. This occurs if the DPL code within a real-time task (e.g. POS0) cannot be completed in time. Use parameter #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.	May	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
65	Invalid function block parameter(s). You have called a FB within a DPL program but one or more of the inputs are invalid.	Yes	Yes	Yes
66	User heap too large. The program has been compiled for a target that has more RAM than this one has. Occurs at startup.	Yes	No	Not Run
67	RAM file does not exist or a non-RAM file id has been specified.	Yes	Yes	Yes
68	The RAM file specified is not associated to an array.	Yes	Yes	Yes
69	Failed to update drive parameter database cache in Flash memory.	Yes	No	Not Run
70	User program downloaded while drive enabled. Will occur if #MM.37 = 1 and a program is downloaded.	May	No	Yes
71	Failed to change drive mode	Yes	No	Yes
73	Fast parameter initialisation failure	Yes	No	No
74	Over-temperature	Yes	Yes	Yes
75	Hardware unavailable. The user program attempted to access unavailable hardware. e.g. if access is made to digital I/O, RS485 port or CTNet on SM-Applications Lite module.	Yes	Yes	Yes
76	Module type cannot be resolved. Module is not recognised.	Yes	No	Not Run
77	Inter-option module comms error with module in slot 1.	Yes	Yes	Yes
78	Inter-option module comms error with module in slot 2.	Yes	Yes	Yes
79	Inter-option module comms error with module in slot 3.	Yes	Yes	Yes

**Table 12.2 SM-Applications Lite Error Codes (Continued)**

<b>Error Code</b>	<b>Reason</b>	<b>Trip Drive?</b>	<b>ERROR task?</b>	<b>Prog Halted?</b>
80	Inter-option module comms error with module unknown slot.	Yes	Yes	Yes
81	APC internal error. See parameter #81.38. This may be caused by one of the following: <ul style="list-style-type: none"><li>• CAM table too small</li><li>• A change of too many CAM segments has occurred at the CAM table input</li><li>• CAM is selected but size is zero</li><li>• CAM absolute mode selected and Reset Index or Reset Position in segment is out of range</li><li>• Slot selected as the reference or feedback does not contain a position option module</li><li>• Attempt to change the Reference source or the Feedback source in more than one task</li></ul>	May	Yes	Yes
82	Communications to drive faulty.	May	Yes	Yes





**MOTEURS LEROY-SOMER 16015 ANGOULÊME CEDEX - FRANCE**

338 567 258 RCS ANGOULÊME  
S.A. au capital de 62 779 000 €

[www.leroy-somer.com](http://www.leroy-somer.com)