Simplex Motion Technical Manual

For following motor models:
SC, SE, SH, SM

About this document
Simplex Motion AB makes no representations or warranties regarding the content of this document. We reserve the right to revise this document any time without notice and obligation.

The document describes the general usages of the following motor models: SCxxx, SExxx, SMxxx and SHxxx (where the x are the motor size designations). For the specific installation, specifications and dimensions of the different motor models, see the motor specifications document for each motor series.

Index

1 Safety

1.1 Qualification of personnel

Only technicians who are familiar with and understand the contents of this manual and the other relevant documentation are authorized to work on and with this drive system. The technicians must be able to detect potential dangers that may be caused by setting parameters, changing parameter values, creating and changing events and generally by the operation of mechanical, electrical and electronic equipment. The technicians must have sufficient technical training, knowledge and experience to recognize and avoid dangers. The technicians must be familiar with the relevant standards, regulations and safety regulations that must be observed when working on the drive system (etc. EMC Directive, Low Voltage directive and Machinery Directive). If the system is used outside the EU, international, national and regional directives must be observed.

1.2 Intended Use

The integrated servomotors systems described here are products for general use that conform to the state of the art in technology and are designed to prevent any dangers. However, drives and drive controllers that are not specifically designed for safety functions are not approved for applications where the functioning of the drive could endanger persons. The possibility of unexpected or unbraked movements can never be totally excluded without additional safety equipment.

For this reason, personnel must never be in the danger zone of the servomotors unless additional suitable safety equipment prevents any personal danger. This applies to operation of the machine during production and also to all service and maintenance work on servomotors and the machine. The machine design must ensure personal safety. Suitable measures for prevention of property damage are also required.

In all cases the applicable safety regulations and the specified operating conditions, such as environmental conditions and specified technical data, must be observed.

The servomotor system must not be commissioned and operated until completion of installation in accordance with the EMC regulations and the specifications in this manual. To prevent personal injury and damage to property damaged servomotors systems must not be installed or operated until this is done.

Changes and modifications of the servomotor systems are not permitted and if made no warranty and liability will be accepted.

The drive systems must not be operated in an environment subject to explosion hazard.

1.3 Hazard Categories

Safety notes and general information are indicated by hazard messages in the manual. In addition there are symbols and instructions affixed to the product that warn of possible hazards and help to operate the product safely. Depending on the seriousness of the hazard, the messages are divided into three hazard categories.

⚠ DANGER

DANGER indicates an imminently hazardous situation, which, if not avoided, will result in death, serious injury, or equipment damage.

⚠ WARNING

WARNING indicates a potentially hazardous situation, which, if not avoided, can result in death, serious injury, or equipment damage

⚠ CAUTION

CAUTION indicates a potentially hazardous situation, which, if not avoided, can result in injury or equipment damage.

1.4 General safety instructions

⚠ DANGER

EXPOSED SIGNALS
Hazardous voltage levels may be present if using an open frame power supply to power the product.

Failure to follow these instructions will result in death or serious injury.

⚠ WARNING

EMERGENCY STOP
If connecting the motor to the machine, build an external emergency stop circuit that immediately stops operation and shuts down power in case of an emergency.

ACCESS TO MOVING PART
Always ensure that no personnel can access the motor before operation as it has accessible moving parts.

LOSS OF CONTROL

- The system manufacturer must take the potential error possibilities of the signals and the critical functions into account to ensure a safe status during and after errors. Some examples are: emergency stop, final position limitation, power failure and restart.
- The assessment of error possibilities must also include unexpected delays and the failure of signals or functions.
- Suitable redundant control paths must be in place for dangerous functions.
- Check that measures taken are effective.

HEAT
The motor will become hot during operation, so do not touch the motor with bare hands. Failure to observe this caution may result in burns.

MODIFICATIONS
Do not attempt to disassemble, repair, or modify the product. Do not change any wiring while power is being supplied.

Failure to follow these instructions can result in death or serious injury

⚠ CAUTION

FAST CHANGES IN MOVMENT
Always attach the motor to a fixed structure before use. Large torques can be generated if target values is changed. The self-weight of the motor is then not enough to hold it stable.

HOT PLUGGING!
Do not connect or disconnect power, logic, or communication while the device is in a powered state. Remove DC power by powering down at the AC side of the DC power supply.

ENVIRONMENT

- Install the servomotor only in environments that meet the requirements for its protection class.
- Do not step on or place a heavy object on the motor. Failure to observe this caution may result in injury.
- Be sure to prevent any foreign objects from entering the product. Failure to observe this caution may result in malfunction or fire.

CABLES
Do not damage the cables or subject them to excessive stress such as bending or stretching. Do not place heavy objects on the cables or the cables between other objects where they might be pinched.

Check the wiring to be sure it has been performed correctly. Connectors and pin layouts are sometimes different for different models. Always confirm the pin layouts in technical documents for your model before operation.

Failure to follow these instructions can result in equipment damage.

2 Communication

The motors have several means of communication for control. Whereas Modbus and CAN are available on most models, USB is limited to the SH and SM series:

Series	Modbus	CAN	USB
SE	Yes (pin 7 & 8)	Yes (pin 5 & 6)	No
SC	Yes (pin 7 & 8)	Yes (pin 7 & 8)*	No
SH	Yes (pin 7 & 8)	Yes (pin 7 & 8)	Yes
SM	Yes (pin 7 & 8)	Yes (pin 7 & 8)	Yes

*CAN is not available on all SC models. Please refer to the technical data sheet for more information

⚠ CAUTION

HOT PLUGGING!
Do not connect or disconnect power, logic, or communication while the device is in a powered state. Remove DC power by powering down at the AC side of the DC power supply.

2.1 Modbus communication

The RS485 Modbus RTU interface is a half duplex master-slave protocol. Up to 32 devices can be connected on the same RS485 lines so that one single master can control up to 31 drive devices. Each device has a unique address 1…126. The default setting is address 1, but it is easily changed by writing to register <Address>. Modbus ASCII mode is not supported.
Baud rate and parity settings are available through the <ModbusControl> register. (SC, SH & SM: Communication, SE: Modbus Control). Default settings are 57600 baud and even parity. The Modbus communication is affected directly when this register is written. To change the settings after the next power cycle, there is a delay that can be enabled.
All registers accessed through the Modbus protocol have 16 bits, and can be both unsigned and signed. Registers that have 32 bits datatypes need dual reads or writes to be accessed. The most significant 16 bits are at the even register address, while the least significant 16 bits are at the odd address.

The following Modbus function codes are supported:

Value	Description
03	Read holding registers
06	Write single register
08	Diagnostics
16	Write multiple registers

For further information on the Modbus standard please consult www.modbus.org
When using a PC computer for control, there are a number of low cost USB-RS485 converters available on the market.

The interface also supports RS232 communication. But the signaling voltages are limited to 0/3.3V, sometimes referred to as RS232 TTL, and do not support the RS232 standard +/-3..12V signaling. However in most cases it is possible to connect to standard RS232 ports using a series resistor of 1kOhm between the master system TX line and the motor unit RX connection. Note that RS232 does not support several slave devices, thus only one motor can be used at a time. When using RS232 the connections are according to:

Pin	Name	RS232 usage
7	IN7/RS485A/ CAN L	RX (receive). Connect to master system TX (transmit) signal. Use a 1kOhm series resistor if standard RS232 signal levels are used.
8	IN8/RS485B/ CAN H	TX. (transmit). Connect to master system RX (receive) signal.

The communication configuration is done using the <ModbusControl> register:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Effect		Delay	Polari-sation	Interface type				Auto Invert		Parity		Baud rate			

Communication speed is set with 'Baud rate':

Value	Baud rate
0	4800
1	9600
2	19200
3	38400
4	57600 (Simplex Motion default)
5	115200
6	Not used
7	Not used

‘Parity’ selects if and how the parity bit is used:

Value	Description
0	Even parity bit (Modbus standard, Default)
1	Odd parity bit
2	No parity bit

‘Auto invert’ automatically changes between standard and inverted (legacy) communication:

Value	Description
0	Off
1	Detects if the communication is inverted and changes between interface type 1 and 2. (default)

The interface is configured according to ‘Interface type’:

Value	Description
0	**Communication interface disabled. Connections used for digital inputs.**
1	**RS485 standard (default)**
2	**RS485 inverted. This is the same as swapping A/B connections. (legacy setting)**
3	**RS232 standard. Signal level is low when idle (0VDC).**
4	**RS232 inverted. Signal level is high when idle (3.3VDC).**
8	CAN communication (for the models that support this).

‘Polarisation’ can be configured to increase resistance to electrical noise:

Value	Description
0	Weak polarization of bus (default)
1	Strong polarization of bus (useful with bus termination, increases tolerance to noise)

‘Delay’ is used to turn Modbus standard delay on and off:

Value	Description
0	Modbus standard delay of 3,5 characters is on
1	Modbus standard delay is off and the motor replies immediately

It is also possible to select when the new communication settings should take effect by the ‘Effect’ bit. This makes it possible to change the changes to the motor before a power cycle.

Value	Description
0	New settings take effect immediately
1	New settings take effect on next power cycling

Some additional notes:

- The RS485 bus signals are denoted A and B. As there are systems using either A or B as the positive signal, in some cases it can be necessary to swap A and B. This product expects A to be the positive signal and B to be the negative. Swapping A and B can also be done by changing the interface settings to RS485 inverted in the <ModbusControl> register.
- Bus polarization is usually needed to define the bus state when no device is transmitting. This device has an internal weak polarization that is sufficient for applications where a termination resistor is not used. It is also possible to enable a strong polarization (should only be enabled on one unit on the bus) for cases when a termination resistor is used.
- A termination resistor (100-120Ohm) is recommended when using a high baud rate (>57600) and long cables (>50m).
- Using the Modbus protocol on some systems shows register numbering with an offset of 1.
- The RS485/RS232 interface is not isolated, so the ground potential must be the same as used for the power supply.
- If using a USB to Serial Port and a high baud rate (115200), the latency timer for the COM port might need to be updated to a lower value then standard settings used by windows (5ms have been tested to work).

The Simplex Motion Tool PC software supports use of the Modbus interface for configuration and testing of the device.

2.2 CAN communication

The CAN interface can be used by external systems to control and communicate with the motors, but it also enables communication between the motors themselves. The CAN protocol is selected by the firmware installed in the motor.

For a description of the available CAN-protocols, please visit the communication section on the Simplex Motion website.

The configuration of CAN is dependent on the motor series.

2.2.1 SC, SH and SM series

CAN pins and configuration register are shared with Modbus, only allowing one protocol to be used at the time.

Pin	Name
7	IN7 / RS485A / CAN L
8	IN8 / RS485B / CAN H

The CAN communication configuration is done using the <Communication> register:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Effect				Interface type								Baud rate			

The interface is configured according to 'Interface type':

Value	Description
0	Communication interface disabled. Connections used for digital inputs.
1	RS485 standard (default)
2	RS485 inverted. This is the same as swapping A/B connections. (legacy setting)
3	RS232 standard. Signal level is low when idle (0VDC).
4	RS232 inverted. Signal level is high when idle (3.3VDC).
8	**CAN communication (for the models that support this**).

Communication speed is set with 'Baud rate':

Value	Baud rate
0	Not used
1	Not used
2	50kbps
3	125kbps
4	250kbps (Simplex Motion default)
5	500kbps
6	800kbps
7	1000kbps

It is also possible to select when the new communication settings should take effect by the 'Effect' bit:

Value	Description
0	New settings take effect immediately
1	New settings take effect on next power cycling

2.2.2 SE series

CAN pins and configuration registers are separate, which allows for Modbus and CAN to be used in parallel.

Pin	Name
5	IN5 / CAN L
6	IN6 / CAN H

The communication configuration is done using the <CANControl> register:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Enable												Baud rate			

Communication speed is set with 'Baud rate':

Value	Baud rate
0	Not used
1	Not used
2	50kbps
3	125kbps
4	250kbps
5	500kbps
6	800kbps
7	1000kbps (Simplex Motion SE-series default)

The CAN interface can be enabled and disabled using the 'Enable' bit:

Value	Description
0	CAN interface disabled.
1	CAN interface enabled (Simplex Motion SE-series default)

2.3 USB communication

(**Note**; Only SH- and SM-Serie motors have a USB interface)
The hardware that have a USB interface, is using the USB type B mini connector type. The interface has full speed (12Mbit/s) and utilized the USB HID protocol (this does not require a custom driver when connecting the device to a PC computer since the operating system provides default support for HID devices).
The device is partly powered by the USB connection; It is possible to communicate with the device and perform configuration without additional power supply. But to start the motor the external power supply is necessary.
If several Simplex Motion devices are connected to the same PC computer, via a USB hub for example, they are separated with unique addresses. This address is the same used for RS485 Modbus communication, and is set in register <Address>.
Please note that the USB interface is not very robust for use in harsh environments and high levels of electrical noise.
It is very important that the USB bus and the motor power supply share the same ground potential, as the USB interface is not isolated. The USB cable length is limited to 5 meters. For applications that require long cables and harsh environments the Modbus interface is recommended.
The Simplex Motion Tool PC software supports use of the USB interface for configuration and testing of the device.

3 Registers

The unit is entirely controlled by its registers that can be read and written using the USB interface, Modbus RTU or the CAN protocol. This chapter lists all available registers as well as an explanation of the different types and how they are stored.

3.1 Register types

3.1.1 Read and Read/Write

The registers in the motor are divided into read only (R) and read/write (R/W). The read only registers contain either static information such as serial number of firmware, or motor process data. The read/write registers contain the parameters for controlling the motor.
NOTE: All the read/write registers are changed instantly when written to. Please be aware of this when writing during operation of the motor.

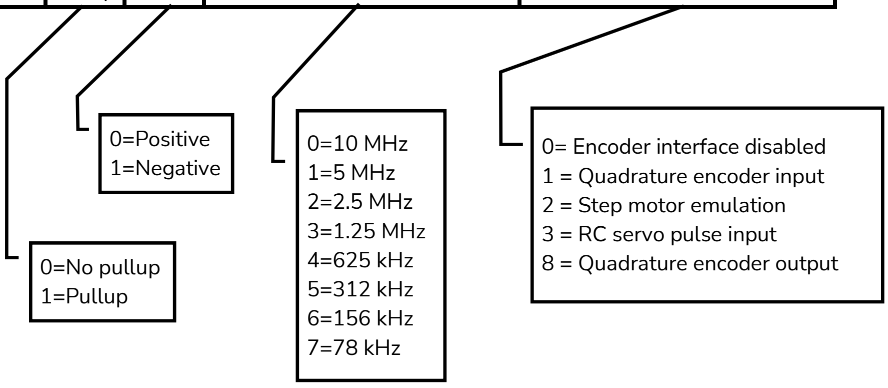
3.1.2 16 and 32 bit

All registers have 16 bits, and can be both unsigned and signed. Parameters that have 32 bits are divided into two consecutive registers with the most significant 16 bits are at the even register address, while the least significant 16 bits are at the odd address.

3.1.3 Single and multi-value registers

There are two variants of the registers used in the motor; single and multi-value
The single value register stores a single parameter value, such as time, temperature or torque.
The multi-value register employ bitfields to store multiple individual values and parameters. Different bits within the bitfield may represent various status flags, parameters, or data values. An example of that is the <180> EncoderControl register:

Bit 15	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
						Pullup	Dir	Encoder filter				Encoder mode			



Example:
Configure Quadrature encoder input with standard filtering and negative rotation:
Encoder mode = 1, converted to binary = 0001
Encoder filter = 4, converted to binary = 0100
Dir = 1
Pullup = 0

Bit 15	**Bit 14**	**Bit 13**	**Bit 12**	**Bit 11**	**Bit 10**	**Bit 9**	**Bit 8**	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
						Pullup	Dir	Encoder filter				Encoder mode			
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**

0000 0001 0100 0001 = 321
<180> EncoderControl = 321

The standard calculator in Windows can be used to convert between decimal and binary. Change the mode to “Programmer” in the top left menu.

3.2 Register memory

There are 3 copies of the entire register map:

Register memory	**Description**
Standard (RAM memory)	At startup the contents in the 'Store' register memory is loaded into the 'Standard' register map in RAM memory. During operation it is always the Standard memory map that is used. It is read and written using the communication facilities. But this memory loses all its contents if the power supply is removed. By use of the 'Store' mode setting it is possible to write this register map to the 'Store' register memory. This way the unit will wake up after next power on with these register contents.
Store (non volatile FLASH memory)	This memory holds the register contents to use at power on startup. It can be written from the 'Standard' register memory by mode 'Store', and it can be written to the 'Standard' memory by the mode 'Reload'.
Factory (non volatile FLASH memory)	This memory holds the factory default register settings. It can't be written by other means than upgrading the firmware. The register settings in this memory can be written to 'Standard' and 'Store' memory by using the 'Factory' mode.

3.3 Register map

Complete register map with all available settings.

Register No.	**Type**	**Name**	**Description**	**Range:**	**R R/W**
1	uns16	VerParameters	Version number for the parameter structure stored internally. Stored as a hexadecimal number, the most significant byte is major revision and the lower is minor. 0x0113 is interpreted as parameter structure version 01.13	0..65535	R
2	uns16	VerFirmware	Version number of the software in the unit. Stored as a hexadecimal number, the most significant byte is major revision and the lower is minor. 0x0206 is interpreted as firmware version 02.06	0..65535	R
3	uns16	VerHardware	Version number of the unit hardware. Stored as a hexadecimal number, the most significant byte is major revision and the lower is minor. 0x2311 is interpreted as Hardware version 23.11	0..65535	R
10-19	string	ModelName	Model name stored as a string. 20 character string stored in 10pcs 16 bit registers.		R
20-29	string	SerialNumber	The unique serial number of this unit stored as a string. 20 character string stored in 10pcs 16 bit registers.		R
30-39	string	UserString1	A user defined string. 20 character string stored in 10pcs 16 bit registers.		R/W
40-49	string	UserString2	A user defined string. 20 character string stored in 10pcs 16 bit registers.		R/W
50	uns16	Address	Unit address used for communication. Needs to be unique when there are several units connected to the same bus or computer (via USB hub or several units on the same Modbus data bus). **Note:** When this value is changed it does not take effect until a power cycling (make sure to write the new setting to memory first), or reset operation has been performed (setting <Mode> register to 1).	1..126	R/W
51	uns16	Identification	Used for secure identification of Simplex Motion units	0..65535	R/W

52	uns16	SC, SH & SM-series: Communication SE-series: Modbus Communication (For CAN interface enable and baud rate, see register <55>)	Sets baud rate and parity settings of the Modbus RTU/CAN interface. Use Bit 15 to choose if the changes should take effect immediately or after next power cycling. See section 2. **Bits** / **Description** 0..3: Modbus baud rate setting: 0 = 4800, 1 = 9600, 2 = 19200, 3 = 38400, 4 = 57600 (Simplex Motion default), 5 = 115200 — CAN baud rate setting: 0 = not used, 1 = not used, 2 = 50kbps, 3 = 125kbps, 4 = 250kbps (Simplex Motion default), 5 = 500kbps, 6 = 800kbps, 7 = 1000kbps 4..5: Parity settings: 0 = Even parity bit (Modbus standard, Default), 1 = Odd parity bit, 2 = No parity bit 6: Not used 7: Auto invert: 0 = Off, 1 = On (default) 8..11: Interface type: 0 = Interface disabled, 1 = RS485 standard (default), 2 = RS485 inverted (legacy), 3 = RS232 standard (idle = low), 4 = RS232 inverted (idle = high), 8 = CAN standard 12: Extra settings: 0 = Strong polarization disabled (default), 1 = Strong polarization enabled (useful with bus termination) 13: Delay: 0 = On, 1 = Off 14: Not used 15: Effect of new settings: 0 = New settings take effect immediately, 1 = New settings take effect on next power cycling	0..65535	R/W
55	uns16	CAN Communication (SE series only)	Sets baud rate and of the CAN interface in the SE-series. See section 2. **Bits** / **Description** 0..3: CAN baud rate setting: 0 = not used, 1 = not used, 2 = 50kbps, 3 = 125kbps, 4 = 250kbps, 5 = 500kbps, 6 = 800kbps, 7 = 1000kbps (Simplex Motion SE series default) 4..14: Not used 15: Enable CAN communication: 0 = CAN communication disabled, 1 = CAN communication enabled (Simplex Motion SE series default)	0..65535	R/W
100	uns16	Supply	Measured supply voltage. Unit is 0.01V. Decimal value 1193 means 11,93V	0..3000	R
101	uns16	TempElectronics	Measured temperature of the electronics. Unit is 0.01°C. Decimal value 2273 means 22,73°C	0..12500	R
102	uns16	TempMotor	Estimated temperature of the motor winding. Unit is 0.01°C. Decimal value 2273 means 22,73°C	0..12500	R

105	uns16	Heating	Used to control heating for low temperature environments. See section 5.2 **Bits** | **Description** 0..3 | Power level: 0 = Disabled 1 = Low power (half of nominal) 2 = Nominal power 3 = High power (double of nominal) 4..5 | Target temperature setting 0 = -40°C 1 = -30°C 2 = -20°C ...		R/W
106	uns16	Brake	Control of external brake. See section 5.3. **Bits** | **Description** 0..1 | Output connected to brake 0 = OUT1 1 = OUT2 2 = OUT3 3 = OUT4 2..4 | Not used 5..7 | Hold force (% of Actuation force) 0 = 12.5% 1 = 25% 2 = 37.5% 3 = 50% 4 = 62.5% 5 = 75% 6 = 87.5% 7 = 100% 8..10 | Actuation force 0 = 12.5% 1 = 25% 2 = 37.5% 3 = 50% 4 = 62.5% 5 = 75% 6 = 87.5% 7 = 100% 11..12 | Control mode 0 = On Mode 1 = On Target 13 | Active low 14 | Active 15 | Brake function 0 = Disabled 1 = Enabled		R/W
107	uns16	Brake timing	Brake timing. See section 5.3. **Bits** | **Description** 0..7 | Engage time 0-255ms 8..15 | Release time 0-255ms	0-255	R/W

110	uns16	Overvoltage	Control the behavior when reaching a set voltage limit. Shut down or sinking of energy through an external resistor. **Bits** / **Description** 0..9 — Voltage limit The limit at which to activate the voltage protection feature. The unit is 0.1V 0 = The default motor maximum voltage rating is used. 10 — Auto Voltage Not implemented yet. 11 — Reserved 12..15 — Mode What should happen the voltage limit is reached. 0 = Turn off motor 1 = Engage external brake resistor on OUT1 2 = Engage external brake resistor on OUT2 3 = Engage external brake resistor on OUT3 4 = Engage external brake resistor on OUT4	0..65535	R/W
120	uns16	SpreadSpectrum	Control of the spread spectrum feature, used to minimize conducted switching noise on the power supply lines. This is accomplished by continuously varying the switching frequency. **Value** / **Description** 0 — Turned off 1 — Frequency variation +/-1.25% 2 — Frequency variation +/-2.5% 3 — Frequency variation +/-5% 4 — Frequency variation +/-10%, Default setting 5 — Frequency variation +/-20%	0..5	R/W
121	uns16	SpeedFilter	Control of motor speed measurement filter. 0 = no filtering. 4 = normal filtering. Increasing value is equal to more filtering. See section 4.2.	0..15	R/W
140	uns16	InputPolarity	The 8 lower bits control input polarity on the inputs IN1-IN7. When set to 0 the corresponding input is active high, while it is active low if set to 1. See section 4.9.1.	0..255	R/W
141	uns16	InputThreshold	Threshold level for low/high for the inputs IN5-8. The 16bit value represents the range 0V to max V. Depending on what model is used, the max V differs, se motor specification for exact value. A typical setting at 1.0V is the value 13107 for a +5V system. See section 4.9.1.	0..65535	R/W
145	uns16	Input	8 bits hold states for digital inputs IN1..7, IN1 in least significant bit. 1 = active input. See section 4.9.1.	0..255	R
150-153	uns16	OutputControl[4]	This register controls the mode of a digital output, allowing simple, pulse, PWM or RC servo pulse output. See section 4.9.2.	0..65535	R/W
160-163	uns16	Output[4]	The 4 output values. These are interpreted differently depending on the output modes set in the respective OutputControl register. See section 4.9.2.	0..65535	R/W
170-173	uns16	Analog[4]	Values from analog inputs AIN1..4. The values are full 16 bits that represent 0 to Max voltage on inputs. Depending on what model is used, the max V differs, se motor specification for exact value. See section 4.9.1. **Nr** / **Description** 170 — AIN1 171 — AIN2 172 — AIN3 173 — AIN4	0..65535	R

180	uns16	EncoderControl	Controls function of quadrature encoder inputs. See section 4.9.3.	0..65535	R/W

Bits	Description
0..3	Encoder mode 0 = disabled 1 = quadrature encoder input 2 = step/direction input interface 8 = quadrature encoder output
4..7	Encoder filter Sets encoder signal filtering 0..7. Default is 4.
8	Invert direction if set to 1
9	Enable pull up resistor if set to 1. There is a weak pull down resistor when set to 0 (default).

Encoder filter values:

Value	Max pulse frequency
0	10 MHz
1	5 MHz
2	2.5 MHz
3	1.25 MHz
4	625 kHz
5	312 kHz
6	156 kHz
7	78 kHz

184/185	int32	Encoder	Value from quadrature encoder interface. Counts 4 * pulse frequency from encoder when using the quadrature encoder mode. In the step/direction mode this register holds the pulse count. See section 4.9.3. Note! Changes to this register is currently not supported over CAN.	-2147483648 .. 2147483647	R/W
200/201	int32	MotorPosition	Current motor position. 4096 positions per revolution. See section 4.2. Note that it is possible to change the motor position resolution, see register MotorOptions.	-2147483648 .. 2147483647	R/W
202	int16	MotorSpeed	Measured motor speed. Unit is positions/second / 16. Value in unit rpm = 60 * MotorSpeed / 256. See section 4.2	-32768..32767	R
203	int16	MotorTorque	Measured motor torque. Unit is 1mNm. See section 4.2	Depends on motor model	R
204	int16	MotorTorqueMax	Setting of torque limit value. Unit is 1mNm. See section 0	Depends on motor model	R/W
205	int16	MotorTorqueStop	Maximum torque to use in quickstop of motor in case of error. See section 4.2	Depends on motor model	R/W
206	int16	MotorVd	Motor flux voltage. Only for debugging purposes	-32768..32767	R
207	int16	MotorVq	Motor torque generating voltage. Only for debugging purposes.	-32768..32767	R
208	uns16	MotorAngle	Motor commutation angle within electrical turn. Only for debugging purposes.	0..65535	R
210	uns16	MotorMagneticA	For debugging		R
211	uns16	MotorMagneticA	For debugging		R
212	uns16	MotorOptions	Misc motor configurations. Can be used to set position sensor resolution. See section 4.2	0..65535	R/W
220	int16	CurrIa	For debugging		R
221	int16	CurrIb	For debugging		R
222	int16	CurrId	Motor flux current for debugging.	-32768..32767	R
223	int16	CurrIq	Motor torque current for debugging.	-32768..32767	R
224	int16	CurrIdFilter	For debugging		R/W
225	int16	CurrIqFilter	For debugging		R/W
226	int16	CurrIdKp	For debugging		R/W
227	int16	CurrIdKi	For debugging		R/W
228	int16	CurrIqKp	For debugging		R/W
229	int16	CurrIqKi	For debugging		R/W
300	int16	RegKp	Regulator proportional parameter. Normal values 500..2000. See section 4.3	0..10000	R/W
301	int16	RegKi	Regulator integrative parameter. Normal values 500..2000. See section 4.3	0..10000	R/W
302	int16	RegKd	Regulator derivative parameter. Normal values 500..2000. See section 4.3	0..10000	R/W
303	int16	RegLimit	Limit value for regulator integration. Normal values 100..500. See section 4.3	0..65535	R/W
304	int16	RegDelay	Controls derivative calculation filtering by setting time delay. Normal values 2..4 Larger values limits the noise, but introduces some time lag. See section 4.3	0..8	R/W
305	int16	RegFriction	Speed feedforward term. Used when friction increases with speed. Unit is Nm/rpm * 10E-6. See section 4.3	0..200	R/W

306	int16	RegInertia	Acceleration feedforward term. Used for high inertia loads. Unit is load inertia, kgm2 * 10E-6. See section 4.3	0..1000	R/W
307	uns16	RegDeadband	Deadband on regulator input error. Typical values 0..20. Higher values reduce motor noise when stationary (regulator hunting) but degrades positioning precision. See section 4.3	0..100	R/W
308	int16	RegError	Regulator error, sometimes called following error. The actual difference between present and target values that are inputs to the regulator. The resolution is 4 times larger than the actual position difference, so the maximum value 8192 corresponds to 2048 positions, or one half shaft revolution. See section 4.3	-8192..8191	R
309	uns16	RegErrorMax	Maximum allowed regulator error. Sets status bit when the error gets beyond this value. Same unit as the RegError register. See section 4.3	0..65535	R/W
310	int16	RegOutput	Regulator output (Torque request). Value is relative to the model maximum torque. Useful for debugging purposes. See section 4.3	0..65535	R
350	int16	RampSpeed	Current speed. Unit is positions/second / 16. Register value = rpm * 4096 / 16 / 60. See section 4.4	0..25600	R
351	int16	RampSpeedMax	Setting of maximum speed. Unit is positions/second / 16. Register value=rpm * 4096 / 16 / 60. See section 4.4	0..25600	R/W
352	int16	RampAcc	Current acceleration. Unit is positions/second^2 / 256. Register value = rpm/s * 4096 / 256 / 60. See section 4.4	0..20000	R
353	int16	RampAccMax	Setting of acceleration value. Unit is positions/second^2 / 256. Register value = rpm/s * 4096 / 256 / 60. See section 4.4	0..20000	R/W
354	int16	RampDecMax	Setting of deceleration value. Unit is positions/second^2 / 256. Register value = rpm/s * 4096 / 256 / 60. See section 4.4	0..20000	R/W
355	int16	RampJerk	Not used at the moment. Will later be implemented to control 3rd derivative of position during ramp control.		R/W

400	uns16	Mode	Controls mode of drive. See section 4.1	0..201	R/W

Value	Name	Description
0	Off	Stop mode, motor is off
1	Reset	Resets all running data and then enters Off mode.
4	Shutdown	When the driver has been shut down because of an error. Motor is off.
5	Quickstop	Motor stopped in a controlled manner, then turned off.
6	Firmware	Firmware upgrade mode. Update firmware through USB or Modbus connection. To exit Firmware mode without upgrading requires a power cycle.
7	Factory	Resets all parameters to factory default settings.
8	Reload	Reloads parameters from non volatile memory and then enters Reset mode.
9	Store	Store current registers to non volatile memory. Then jumps to previous mode.
10	Pwm	PWM mode, open loop control.
19	Freewheel	Freewheeling up to set max speed
20	Position	Closed loop control of position. **NOTE!** This mode has no ramping control, and is intended for cases when the target value is changed in small steps, where an external system handles speed and acceleration limits. For example using encoder or step/dir inputs. **Too large changes in position using this mode can damage the motor!**
21	PositionRamp	Closed control of position with ramp control.
23	Rotary	Closed control of position with ramp control. **NOTE!** Make sure <TargetFilter> is set to 0 for full functionality
32	Speed	Speed control mode. **NOTE!** This mode has no ramping control, and is intended for cases when the target value is changed in small steps, where an external system handles speed and acceleration limits. **Too large changes in speed using this mode can damage the motor!**
33	SpeedRamp	Speed control mode with ramp control.
34	SpeedLow	A special low speed mode for higher resolution at low speed. The supplied target is divided by 256 before generating the actual speed value.
35	SpeedLowRamp	Low speed mode with ramping control.
40	Torque	Torque control mode.
50	SeqPosRamp	Sequence position control by ramping.
51	SeqPosInterpolate	Sequence position control with interpolation.
54	SeqPosFinished	Sequence position control finished.
55	SeqSpeedRamp	Sequence control of speed by ramp control.
56	SeqSpeedInterpolate	Sequence control of speed by interpolation.
59	SeqSpeedFinished	Sequence control of speed is finished.
60	Beep	Motor produces sound at 500Hz.
70	Homing	Initiates motor homing.
200	DemoOn	Starts demo mode. Uses a potentiometer connected to IN2 and a pushbutton connected to IN1.
201	DemoOff	Stops demo mode.

401	uns16	ModeStartup	Set the value of <Mode> at power on. Allows automatic start at power on.	0..201	R/W

410	uns16	Status	Drive status. Each bit has status information according to the table below. This status word is used for several things, it can trip the driver or start recording data. The bits are only active while the condition is true. See section 6.	0..65535	R

Bit	Name	Description
0	Fail	Internal error in the driver.
1	Communication	Communication error.
2	Current	Hardware overcurrent protection triggered. Motor current beyond normal values.
3	Voltage	Input voltage is too high or low. Voltage < 10V or Voltage > 55V.
4	Temperature	Temperature of drive is too high, motor temp > 120°C or electronics temp > 100°C.
5	Torque	Motor torque limit active.
6	Locked	Locked shaft condition detected. Torque > 10% of rated, but speed < 1rps.
7	Regulator	The regulator has a large error. RegError > RegErrorMax register value.
8	Moving	Motor is rotating, Speed > 0.1rps.
9	Reverse	Motor is rotating in reverse direction.
10	Target	Target reached when ramping position control.
11	Reserved	For future use.
12	InputA	Digital input, specified by the <StatusInputs> register.
13	InputB	Digital input, specified by the <StatusInputs> register.
14	User1	For user application, set by event handler.
15	User2	For user application, set by event handler.

411	uns16	StatusLatched	A latched version of the <Status> register. The corresponding bit in this register is set when it is set in the <Status> register, and then kept set until it is read by the user. See section 6.	0..65535	R
412	uns16	StatusInputs	This register defines two digital inputs that are available in the status register as InputA and InputB. This is useful for Limit switches that should cause a driver shutdown. It is also possible to filter these inputs from noise. See section 6.	0..65535	R/W

Bits	Description
0..3	Input number to use for InputA
4..7	Input number to use for InputB
8..15	Filter value. 0 = no filtering. Increasing values causes more filtering and larger delay.

413	uns16	MaskQuickstop	Mask to select status bits to cause driver quickstop. A one in a bit enables the corresponding status bit as a trigger for quickstop. A quickstop event causes the motor to stop in a controlled fashion, usually by applying the <RampDecMax> setting for deceleration of the motor speed. See section 6.	0..65535	R/W
414	uns16	MaskShutdown	Mask to select status bits to cause a driver shutdown. A one in a bit enables the corresponding status bit as a trigger for shutdown. A shutdown event disconnects the motor from the driver immediately, causing the motor to run freely from its inertia to a stop. See section 6.	0..65535	R/W
415	uns16	Error	This register holds the latest generated error code. See 6.1 for error codes.	0..65535	R
416	uns16	StopConfig	Configure how motor operates at stop and error. See section 6.		R/W
420/421	uns32	Time	Tracks time as 2000 counts per second. Wraps around after about 12 days. This register can also be written.	0 .. 4294967295	R/W
448/449	int32	TargetRelative	Value is added to <TargetInput>. Used to make a relative change to position or speed. See section 4.3.2.	-2147483648 .. 2147483647	R/W
450/451	int32	TargetInput	Target value for regulator. Used when <TargetSelect> = Register. See section 4.3.2.	-2147483648 .. 2147483647	R/W

452	uns16	TargetSelect	Sets the target source. See section 4.3.2.	0..6	R/W

Value	Name	Description
0	Register	Target is set by a register content. Register <TargetInput> is used as target.
1	Analog 1	Analog value from AIN1 is used as target. Value 0..65535
2	Analog 2	Analog value from AIN2 is used as target.
3	Analog 3	Analog value from AIN3 is used as target.
4	Analog 4	Analog value from AIN4 is used as target.
5	Encoder	Encoder interface is used for target values. The encoder can be set for quadrature encoder input or Step/Dir interface for step motor emulation. The target value is taken from the <Encoder> register.
6	Pulse	A digital input pulse length is used to set target values. Compatible with RC servo pulses. Not yet implemented.

453	int16	TargetMul	Value to multiply with input target value before used by the regulator. See section 4.3.2.	-32768 .. 32767	R/W
454	int16	TargetDiv	Value to divide the input target value with before it is used by the regulator. See section 4.3.2.	-32768 .. 32767	R/W
455	int16	TargetOffset	Value to add to the input target before it is used by the regulator. The Offset is applied after <TargetMul> and <TargetDiv>. See section 4.3.2.	-32768 .. 32767	R/W
456/457	int32	TargetMin	Minimum value for target value. See section 4.3.2.	-2147483648 .. 2147483647	R/W
458/459	int32	TargetMax	Maximum value for target value. See section 4.3.2.	-2147483648 .. 2147483647	R/W
460	int16	TargetHysteresis	Hysteresis value to remove noise from target values. This is typically useful when the target source is an analog input. Applied after Mul/Div/Offset. Typical values 0..1000. See section 4.3.2.	0..65535	R/W
461	uns16	TargetFilter	Allows filtering of target values to reduce noise and limit rate of change. 0 = no filtering, increasing values allows more filtering. Typical values 0..7. See section 4.3.2.	0..16	R/W
462/463	int32	TargetPresent	The current target value as it is sent to the regulator. Useful for debugging. See section 4.3.2.	-2147483648 .. 2147483647	R
470/471	int32	RotaryStart	In Rotary mode, this is the minimum value for register <MotorPosition>. When <MotorPosition> goes below <RotaryStart>, it wraps around to <RotaryStop> – 1. See section 4.3.2.	-2147483648 .. 2147483647	R/W
472/473	int32	RotaryStop	In Rotary mode, this is the maximum value for the <MotorPosition>. When <RotaryStop> is reached, <MotorPosition> wraps around to <RotaryStart>. The maximum value of <MotorPosition> is <RotaryStop> – 1. See section 4.3.2.	-2147483648 .. 2147483647	R/W
480-483	uns16	HomeSequence[4]	Sequence definition for homing sequence. 4 individual sequence steps. The homing features are used to find a position reference at system startup. See more in section 4.6.	0..65535	R/W
490	int16	HomeOffset16	New position value the homing position should have in relation to the 0-position. Only used for legacy reasons. Use HomeOffset32 instead	-32768..32767	R/W
491	uns16	HomeSpeed	Reference speed to use for homing. Unit is positions/second / 16. Register value = rpm * 4096 / 16 / 60. See section 4.6.	0..25600	R/W
492	uns16	HomeAcc	Homing acceleration. Unit is positions/second^2 / 256. Register value = rpm/s * 4096 / 256 / 60. See section 4.6.	0..20000	R/W
493	uns16	HomeTorque	Torque limit to use by hard stop homing. Unit is mNm. See section 4.6.	0..2000	R/W
494	uns16	HomeDoneMode	Mode to switch to when homing sequence is finished. This value is then written to register 400. See section 4.6.	0..201	R/W
495	int16	HomeChange	The difference between the zero positions before and after a completed homing. Useful for debugging and basically shows how much repetitive homings deviate. See section 4.6.	-32768..32767	R
496/497	Int32	HomeOffset32	New position value the homing position should have in relation to the 0-position. Allows the 0 position to be placed at an offset from the homing sensor. See section 4.6.	-2147483648 .. 2147483647	R/W

500	uns16	SeqControl	.Sequence control. See section 4.5.		R/W

Bits	Name	Description
0	Delay mode	When delay mode = 0, timer set by <SeqTime[]> starts immediately. When <SeqTime[]> is up, the function moves to the next sequence entry, even if the set position isn't reached. When delay mode = 1, the timer starts when the set position is reached.
1-2	Time Scale	Sets the time scale of the timer used in delay mode. Bit 1 and 2 = 0, time in milliseconds Bit 1 = 1, bit 2 = 0, time in seconds Bit 1 = 0 and bit 2= 1, time in minutes.
3	Repeat	When Repeat = 0, the sequence will end when encountering <SeqTime[]> = 0, When Repeat = 1 the sequence will continue regardless of <SeqTime[]> = 0.

501	uns16	SeqIndex	Current index into the table of positions and time values. Can have values from 0 to 15. During sequence control this register automatically increments for each processed table entry. See section 4.5.	0..15	R/W
510-540	int32	SeqTarget[16]	Target values of the table. Position: Unit is 1/4096 revolution Speed: Unit is positions/second / 16 See section 4.5.	-2147483648 .. 2147483647	R/W
570-585	uns16	SeqTime[16]	Time values of the table. Unit is milliseconds, seconds or minutes. See section 4.5.	Mode 50: 0..65535 Mode 51: 0..3999	R/W
600	uns16	ApplControl	Control for custom application code loaded into firmware in the device.	0..65535	R/W
601	uns16	ApplStatus	Status information from custom application code loaded into the firmware.	0..65535	R
602	uns16	ApplRuntime	Runtime indication for custom application code. Indicates the percent of available runtime that is used up by the application code.	0..100	R
603	uns16	ApplVersion	Version of the custom application code. The most significant byte is major revision and the lower is minor.	0..65535	R
620-627	int16	ApplData[8]	8 registers of general use for the custom application.	-32768..32767	R/W
640-647	uns16	Debug[8]	8 registers used for debugging of the custom application.	0..65535	R
680-699	uns16	EventControl[8]	Control register for event. Events are used to cause simple actions to happen from trigger conditions. For example, to set a certain register value when a digital input is activated from a pushbutton or activate an output when a register value is above a certain threshold.	0..65535	R/W

Bits	Description
0..3	Trigger operation Used to determine if trigger condition is met.
4..7	Trigger filter Allows filtering of trigger condition.
8..11	Trigger type 0 = Active, 1 = Edge, 2 = Repeat.
12..15	Data operation Used to manipulate register when event is executed.

See section 4.7 for more information.

700-719	uns16	EventTrgReg[8]	Trigger register number.	0..65535	R/W
720-739	uns16	EventTrgData[8]	Trigger data value. 16-bit value to use with trigger register and operator.	0..65535	R/W
740-759	uns16	EventSrcReg[8]	Source register number.	0..65535	R/W
760-779	uns16	EventSrcData[8]	Source data value. 16-bit value to use with source register and operator.	0..65535	R/W
780-799	uns16	EventDstReg[8]	Destination register to write event result to.	0..65535	R/W
900	uns16	RecState	State of the recorder. The recorder is used to store measurements in a rapid pace for debugging and inspection of dynamic behavior. There is space for 500 measurements of 4 channels, each being 16 bits wide.	0..3	R/W

Value	Name	Description
0	Idle	Recorder in idle, not used.
1	Continuous	Recording continuously
2	Single	Perform one complete recording of 500 values.
3	Trigger	Trigger enabled, recording started when trigger condition met.

901	uns16	RecTrigger	Trigger word. This word is used as a mask with the status register. When an active status bit corresponding to an active <RecTrigger> bit appears the trigger condition is met.	0..65535	R/W
902	uns16	RecPeriod	Sets the recording speed as number of regulator cycles between recordings. Setting this value to 0 provides the fastest possible recording speed, taking all 500 measurements in exactly 0.25s.	0..65535	R/W
903	uns16	RecPreceding	Sets the number of samples to appear before trigger. This feature makes it possible to measure just prior to trigger condition.	0..1000	R/W
904	uns16	RecOffset	Offset position into data for start. Since the data area is used as a circular buffer that runs continuously, the first data point is not always in the first memory position. Instead, the first data is at the <RecOffset> position.	0..999	R
905-908	uns16	RecRegister[4]	Register numbers for the 4 channels to record.	0..4999	R/W
1000-1499	int16	RecData1[500]	Data for recording channel 1. Data can be uns16 or int16 depending on the source register. 500 values in consecutive register addresses.	0..65535	R
2000-2499	int16	RecData2[500]	Data for recording channel 1. Data can be uns16 or int16 depending on the source register. 500 values in consecutive register addresses.	0..65535	R
3000-3499	int16	RecData3[500]	Data for recording channel 1. Data can be uns16 or int16 depending on the source register. 500 values in consecutive register addresses.	0..65535	R
4000-4499	int16	RecData4[500]	Data for recording channel 1. Data can be uns16 or int16 depending on the source register. 500 values in consecutive register addresses.	0..65535	R

4 Device operation

This chapter explains the operation of the unit and how it is controlled through the registers.

4.1 Operating modes

The <Mode> register controls the overall behavior of the motor unit. The following modes are available:

Name:	Value:	Description:
Off	0	Stop mode, motor is off.
Reset	1	Resets drive All running data is reset, such as position. Automatically changes <Mode> to Off mode.
Shutdown	4	The driver is shut down because of an error. Motor is off. This happens if any status bits enabled by the <MaskShutdown> becomes active. This is a feature to shutdown the motor in case of events such as high temperature, internal error etc.
Quickstop	5	Motor stopped in a controlled manner, then turned off. A quickstop event causes the motor to stop in a controlled fashion, by applying the <MotorTorqueStop> braking torque for deceleration of the motor speed. This happens if a status bit enabled by the corresponding bit in the <MaskQuickstop> becomes active.
Firmware	6	Firmware upgrade mode. Causes control to be passed to a bootloader to receive new firmware through the USB or Modbus connection. A special PC software is needed to download the new firmware.
Factory	7	Resets all parameters to factory default settings. Then sets <Mode> to Reset mode.
Reload	8	Reloads the last stored parameters from non volatile memory and resets all running data. This is equivalent to cycling the power supply to restart the unit.
Store	9	Store the current values in the registers to non volatile memory After the register values have been stored, the mode changes automatically to the previous mode.
Pwm	10	PWM mode, open loop control The <Target> value [-32768..32767] is directly converted to motor voltage, where -32768 is full speed reverse, 0 is standstill, and 32767 is full speed forwards. There is no regulator involved, and no ramping. Torque is not limited. This mode is mainly intended for testing and has limited use.
Freewheel	19	Freewheeling mode. Sets the motor in a state where it can rotate freely. If a brake is enabled, it will also be released. If the speed exceeds the set target value, the motor will use the set MotorTorqueMax to slow down.
Position	20	Closed loop control of position This mode uses the PID regulator to perform closed loop regulation of the motor position. Torque limit is active. **NOTE!** This mode has no ramping control, and is intended for cases when the target value is changed in small steps, where an external system handles speed and acceleration limits. For example using encoder or step/dir inputs. **Too large changes in position using this mode can damage the motor!**
PositionRamp	21	Closed loop control of position with ramp control Similar to the 'Position' mode but does also support ramping control of the position. This means controlled acceleration and speed according to user settings. This is the preferred mode since it typically limits torque and supply currents and causes even motions with less vibration.
Rotary	23	Closed control of position with ramp control. <MotorPosition> is limited between <RotaryStart> and <RotaryStop>. When reaching the limits <MotorPosition> will wrap around. Useful when there is a need to keep track of position during a single turn. **NOTE!** Make sure <TargetFilter> is set to 0 for full functionality
Speed	32	Speed control mode Precise speed control down to 0 rpm. **NOTE!** This mode has no ramping control, and is intended for cases when the target value is changed in small steps, where an external system handles speed and acceleration limits. **Too large changes in speed using this mode can damage the motor!**
SpeedRamp	33	Speed control mode with ramp control This is the recommended mode for general speed control applications.
SpeedLow	34	Low speed mode A special speed mode for higher resolution at low speed. The supplied target is divided by 256 before generating the actual speed value.
SpeedLowRamp	35	Low speed mode with ramping control.
Torque	40	Torque control mode Has a speed limit feature as well (set maximum speed in the <RampSpeedMax> register). The required torque <TargetInput> is scaled so that a signed 16 bit value covers the motor maximum torque range. So the maximum torque value is +/-32767.
SeqPos	51	Sequence control of position by ramp control
SeqPosInt	52	Sequence control of position by interpolation
SeqPosFin	54	Sequence control of position is finished.
SeqSpeed	55	Sequence control of speed by ramp control.
SeqSpeedInt	56	Sequence control of speed by interpolation.
SeqSpeedFin	59	Sequence control of speed is finished.
Beep	60	Motor produces sound at 500Hz Target value sets amplitude. Can be used for user communication.
Homing	70	Implements motor homing. Setting this mode starts the homing sequence. Once finished the <Mode> register is set to the contents in the <HomeDoneMode> register.

DemoOn	200	Starts demo mode. Uses a 10k potentiometer connected to +5V/IN2/GND and a pushbutton connected from IN1 to GND for user control. The demo mode uses the potentiometer to set target value, and a pushbutton to change between 4 testmodes. Each press advances the testmode one step, while pressing for more than 1 second jumps to the first testmode. Nr | Testmode | Potentiometer range 1 | Speed regulation | Speed from 0 to 5000rpm. 2 | Low speed regulation | Speed from 0 to 20rpm. 3 | Position regulation | Position from 0 to 8192 (2 turns) 4 | Position regulation with ramping | Position from 0 to 65535 (16 turns)
DemoOff	201	Stops demo mode Changes <Mode> to 'Reset' mode after turning the demo mode off.

4.2 Motor data

Registers related to motor data.

Name	Type	Nr	Description
TempElectronics	uns16	101	Measured temperature of the PCBA. Unit is 0.01°C Overtemperature condition is reflected in a status bit (see section 0).
TempMotor	uns16	102	Estimated temperature of the motor. Unit is 0.01°C The value is calculated by use of a thermal model of the device. The temperature of the electronics is directly measured, and since the electronics have a tight thermal coupling to the device enclosure, and the thermal resistance from motor winding to the enclosure can be described by a simple relationship to motor speed, it is possible to get reasonably accurate results this way. Overtemperature condition is reflected in a status bit (see section 0).
SpeedFilter	uns16	121	Control of motor speed measurement filter. 0 = no filtering. 4 = normal filtering. Values 0..15 are possible. Increasing value is equal to more filtering, which produces a less noisy speed measurement, but at the same time increases time lag in the measurement.
MotorPosition	int32	200/201	Current motor position. 4096 positions per revolution. The value is reset to zero at start. It can be changed by the user by writing the register. The value is reset in the operating mode 'Reset'. Note also that the resolution of the motor position value can be changed with the <MotorOptions> register.
MotorSpeed	int16	202	Measured motor speed. Unit is positions/second / 16. The maximum speed of 6000rpm equals a speed value of 25600. Negative speed values represent rotation in the negative direction. The measured speed is filtered to minimize noise, and the filter is configurable by use of register <SpeedFilter>. rpm = <MotorSpeed> * 16 * 60 / 4096
MotorTorque	int16	203	Measured motor torque. Unit is mNm. Negative values represent torque in negative rotational direction.
MotorTorqueMax	int16	204	Setting of torque limit value. Unit is mNm. Maximum value is 2000, equal to 2.0Nm. Limiting torque to a value that is suitable for the application is recommended. Torque limiting is always active, independent of operating mode. The same limiting value is used for both braking and driving, and in both rotational directions.
MotorTorqueStop	int16	205	Setting of torque limit value for motor stopping in quickstop mode. Unit is mNm. Maximum value is 2000, equal to 2.0Nm.
MotorVq	int16	207	Motor voltage value. Possible values are -32768..0 for negative voltage, and 0..32767 for positive voltage. Mainly used for debugging purposes.
MotorOptions	uns16	212	Miscellaneous motor settings. Bit 12-15 set the motor position resolution according to the following table:

Value bit 12-15	Description
0	12bits, 4096 counts per revolution. Default.and recommended.
1	13bits, 8192 counts per revolution.
2	14bits, 16384 counts per revolution.

Note that the default 12bit resolution offers noise free position values, increasing the resolution will introduce noise into the position value as presented in the <MotorPosition> register.

Name	Type	Nr	Description
CurrIq	int16	223	Motor current value. Possible values are -32768..0 for negative current, and 0..32767 for positive current. Mainly used for debugging purposes.

4.3 PID controller

For regulation of motor speed or position there is a PID controller. This controller calculates the error as the difference between the target value and the present value, and then forms the motor control value as a sum of a proportional, integral and derivative component of the error.
These 3 components have their own gain parameters that set the controller characteristics.

NOTE: The preset values of the PID controller are only intended for general use of the motor. The PID controller values need to be adjusted and optimized for the specific motor application.

For general information on PID controllers, see: http://en.wikipedia.org/wiki/PID_controller

The PID regulator compensates for power supply variation, which means that the same performance of the motor can be expected across the entire power supply range. An exception to this is when the unit is used at high rotational speeds, as the top speed is limited by the supply voltage.

Regulator parameters:

Name	Type	Nr	Description
RegKp	int16	300	Regulator proportional parameter. Normal values 100..1000. This parameter is usually the most important contributor to controller performance, as it sets the 'stiffness' of the motor (resistance to errors). Large values can result in significant overshoot and instability. Low values causes larger positioning errors.
RegKi	int16	301	Regulator integral parameter. Normal values 100..1000. The integral part is responsible for eliminating the residual steady state error of the controller. Large values will affect stability. Low values can cause a stationary error.
RegKd	int16	302	Regulator derivative parameter. Normal values 100..1000. The derivative component slows the transient response and thus helps keep the controller stable and minimizes controller overshoot. This controller part has the largest amount of noise, using the <RegDelay> parameter can minimize this issue. Noise in the controller can show up as audible noise from the motor. Usually it is best to start tuning the PID regulator with this value set to 0.
RegLimit	int16	303	Limit value for regulator integration. Normal values 100..500. When a large error is present for some time the integral part of the controller can become very large and this can cause extreme overshoots of the controller. Therefore there is a limit to the integral of the error, adjustable by this register.
RegDelay	int16	304	Controls derivative calculation filtering, which also produces a delay. Normal values 2..4, range 0..8. Large values will decrease the noise in the derivative component of the regulator, but at the same time increase time lag.
RegFriction	int16	305	Speed feedforward term. Used when friction increases with speed. Setting this parameter correctly greatly reliefs the PID controller and thus decreases the controller error. Unit is Nm/rpm * 10E-6
RegInertia	int16	306	Acceleration feedforward term. Used for high inertia loads. Setting this parameter correctly greatly reliefs the PID controller and thus decreases the controller error. Unit is load inertia, kgm^2 * 10E-6
RegDeadband	uns16	307	Dead band on regulator input error. When the motor is stationary in position regulation mode it is common to hear some audible motor noise. This comes from the constant regulation to stay at the target position, sometimes called 'regulator hunting'. If positioning precision can be allowed to degrade somewhat it is possible to get rid of this noise. By setting a dead band the regulator will not care about errors less than this dead band value, and thus the regulator will be idle. Typical values 0..20. 0 = turn off dead band feature.
RegError	int16	308	Regulator error, sometimes called following error. This value is the calculated controller error. Observing this value lets the user measure the performance of the motor drive unit. It is a good indicator of controller performance when tuning the regulator parameters. The resolution is 16 times larger than the actual position difference, so the maximum value 32767 corresponds to 2048 positions, or one half shaft revolution.
RegErrorMax	uns16	309	Maximum allowed regulator error. Sets status bit 'Regulator' when the error gets beyond this value. This can be used to monitor if the regulator error has been beyond a certain value during a session. Or to shut down the unit if error gets really large. Same unit as register <RegError>.
RegOutput	int16	310	Regulator output (Torque request). Value is signed 16bits relative to the model maximum torque. Useful for debugging purposes.

4.3.1 Feed forward

In some cases, the motor speed and/or acceleration is known, and this makes it possible to help the PID controller by introducing feed forward components. One such case is when running ramp controlled moves, where the target acceleration and speed is continuously calculated. If characteristics of the motor load is known, it is possible to make use of this information for improved control. There are two feed forward components, one for speed and one for acceleration.

The speed feed forward term is used to compensate for loads where the torque increases with rotational speed. The register used is <RegFriction>, and the unit is Nm/rpm * 10E-6. This value is difficult to calculate, so usually experimenting will be necessary. A good start value can be 100.
The acceleration feed forward term compensates for the load inertia, as the torque needs to be increased to change the rotational speed. This is especially important in high inertia applications, such as linear positioning devices with heavy loads. This value can usually be calculated but experimenting can also be used to find an appropriate value. The register used is <RegInertia> and the unit is load inertia (as seen on the motor shaft) kgm2 * 10E-6.

To test and trim the feed forward components one can briefly disable the regulator by setting the PID controller parameters (RegKp, RegKi, RegKd) to zero, and apply a ramp controlled position move. By observing the regulator error across the movement (by using the recorder, see 0) one can change the parameters until the error is minimized.
There is a feature in the Simplex Motion Tool PC software to aid in this tuning.

4.3.2 Target value

The target value is the PID controller setpoint value. It can be obtained from several different sources, configured by the register <TargetSelect>:

Name	Value	Description
Register	0	Target is set by a register content. Register <TargetInput> is used as target. This setting is typically used when the device is continuously controlled through the communication bus.
AIN1	1	Analog value from IN1 is used as target. The analog value has the range 0..65535. This makes setting of the target value by a potentiometer possible. Connect the potentiometer as a resistive divider between the supplied +3.3V or +5V (depending on model) and GND. Any other voltage source providing a 0..+3.3V or 0..+5V voltage can be used.
AIN2	2	Analog value from IN2 is used as target.
AIN3	3	Analog value from IN3 is used as target.
AIN4	4	Analog value from IN4 is used as target.
Encoder	5	The encoder input is used as target. The encoder output is available in register <Encoder>, and this value is used as target value. The encoder interface can be configured both for quadrature encoder input, for step/direction signal interface and RC pulse inputs This feature makes it easy to track another motor that supplies an encoder output, or to emulate a step motor interface.

A few more settings are available for the handling of target values. Scaling and offsetting of target values is of great use when the target source is some external input such as an analog input. The registers <TargetMul> and <TargetDiv> is used for scaling, and the <TargetOffset> for offsetting. The offset is applied after the multiplication and division operations.
It is possible to limit target values by min and max bounds. This is done by the <TargetMin>/<TargetMax> registers.
There are also features to deal with noise on the input target values. This can be done in two ways, by hysteresis or by filtering. The <TargetHysteresis> register allows the target value to change by small amounts, less than the register value, before the actual used target changes. Setting the register to zero eliminates this feature. The <TargetFilter> register allows filtering instead. A value of zero disables the filter, while an increasing value adds more filtering.
For debugging, the final target value as sent to the PID regulator, can be read from the register <TargetPresent>.

A full summary of target related registers:

Name	Type	Nr	Description
TargetRelative	Int32	448/449	Value to add to <TargetInput>. Used to make a relative change to position or speed.
TargetInput	int32	450/451	Value to be used as target input. Used as target source when <TargetSelect> = Register.
TargetSelect	uns16	452	Sets the target value source according to: <table><tr><th>Value</th><th>Name</th><th>Description</th></tr><tr><td>0</td><td>Register</td><td>Target value from register <TargetInput.></td></tr><tr><td>1</td><td>Analog 1</td><td>Analog value from AIN1 is used as target.</td></tr><tr><td>2</td><td>Analog 2</td><td>Analog value from AIN2 is used as target.</td></tr><tr><td>3</td><td>Analog 3</td><td>Analog value from AIN3 is used as target.</td></tr><tr><td>4</td><td>Analog 4</td><td>Analog value from AIN4 is used as target.</td></tr><tr><td>5</td><td>Encoder</td><td>Encoder interface is used, enabling both quadrature encoder input, step/direction signals and RC pulse inputs.</td></tr></table>
TargetMul	uns16	453	Value to multiply with input target value before used by the regulator.
TargetDiv	uns16	454	Value to divide the input target with before it is used by the regulator.

TargetOffset	uns16	455	Value to add to the input target before it is used by the regulator.
TargetMin	int32	456/457	Minimum value for target value
TargetMax	int32	458/459	Maximum value for target value
TargetHysteresis	int16	460	Hysteresis value to remove noise from target values. This is typically useful when the target source is an analog input. Applied after Mul/Div/Offset. Typical values 0..1000.
TargetFilter	uns16	461	Allows filtering of target values to reduce noise and limit rate of change. 0 = no filtering, increasing values allows more filtering. Typical values 0..7. NOTE! When <TargetInput> is set digitally, or Rotary mode is used, the filtering should be set to 0.
TargetPresent	int32	462/463	The current target value as it is sent to the regulator.
RotaryStart	int32	470/471	In Rotary mode, this is the minimum value for register <MotorPosition>. When <MotorPosition> goes below <RotaryStart>, it wraps around to <RotaryStop> – 1.
RotaryStop	int32	472/473	In Rotary mode, this is the maximum value for the <MotorPosition>. When <RotaryStop> is reached, <MotorPosition> wraps around to <RotaryStart>. The maximum value of <MotorPosition> is <RotaryStop> – 1.

4.4 Ramping control

In most applications it is desirable to limit acceleration and speed values to configurable levels. This is accomplished by ramping control. It is available both for speed control and for position control. The mode setting (see 4.1) determines if it is being used or not. Ramping limits are divided in two registers, one for acceleration and one for deceleration. The advantage of separating acceleration and deceleration is that applications with large inertia loads may need low deceleration rates to limit the overvoltage created when the energy from the mechanical load is transferred to the power supply (the motor acts as a generator).

The following table summarizes the available registers for ramping control:

Name	Type	Nr	Description
RampSpeed	int16	350	Current speed. Unit is positions/second / 16. Values 0..25600. This value changes continuously during acceleration/deceleration to reflect the current target speed. It is also used to implement the speed feed forward component of the PID regulator. Register value = rpm * 4096 / 16 / 60
RampSpeedMax	int16	351	Setting of maximum speed. Unit is positions/second / 16. This is the speed limit for speed control mode, and the top speed used for position moves in position control mode. Register value = rpm * 4096 / 16 / 60
RampAcc	int16	352	Current acceleration. Unit is positions/second^2 / 256. This value reflects the present acceleration. Used by the acceleration feed forward component of the PID regulator. Register value = rpm/s * 4096 / 256 / 60.
RampAccMax	int16	353	Setting of acceleration value. Unit is positions/second^2 / 256. Register value = rpm/s * 4096 / 256 / 60.
RampDecMax	int16	354	Setting of deceleration value. Unit is positions/second^2 / 256. Register value = rpm/s * 4096 / 256 / 60.

4.5 Sequence control

To simplify sequence of movements there is a sequence feature. This feature is based on a table of 16 entries with position/speed and time values. The table resides on registers <SeqTarget0> - <SeqTarget15> and <SeqTime0> - <SeqTime15>. Values in <SeqTargetX> registers are standard motor position values (4096 per revolution) or standard speed values (Positions/second / 16), and time values in <SeqTimeX> are in units defined by <SeqControl> bit 1-2 (mode 51 is always milliseconds). The index of the current table entry is stored in register <SeqIndex>.
When a table entry has been processed, the <SeqIndex> register is incremented, and the next table entry is processed. When the last table entry has been processed, the index pointer will wrap-around to 0 to start over on top of the table.
Note: Sequence control of speed is only available from firmware revision 02.00 and later.

This feature can operate in two modes:

Mode	Description
Ramping control sequence	A ramping movement using the Ramping Control parameters to the specified position and the specified time is used for delay until next movement. The timer starts at the beginning of the movement or when the position is reached. When delay mode = 0, timer set by <SeqTime[]> starts immediately. When <Seqtime[]> is up, the function moves to the next sequence entry, even if the set position isn't reached. When delay mode = 1, the timer starts when the set position is reached. The time scale can be set to milliseconds, seconds or minutes: Bit 1 and 2 = 0, time in milliseconds Bit 1 = 1, bit 2 = 0, time in seconds Bit 1 = 0 and bit 2= 1, time in minutes. When Repeat = 0, the sequence is terminated by a time value of 0 in a table entry. When Repeat = 1, the sequence will repeat regardless if there is a <SeqTime[]> = 0. The sequence is also terminated if the <Mode> register is changed.
Interpolation control	The motor performs interpolation of the target value between tables entries. This mode is for continuous motion without stops. For each table entry: A movement is started from the current table entry position towards the next table entry position, to finish after the time specified in the current table entry. Each such movement will have constant speed. In future implementation there will be possibilities to also interpolate speed to have constant acceleration. This mode can be used with a master system continuously writing new values to the table as they are consumed, and in such a way implement continuous custom motion with the limited bandwidth of the communication to the motor. By reading the <SeqIndex> the master can tell what table entries has already been consumed and then write new values to those positions. The table pointer will wrap around from last to first table entry if no 0 value in the time entry has been encountered.

The following <Mode> values are used to control this feature:

<Mode> register value	Description
50	Sequence control of position by ramp control
51	Sequence control of position by interpolation
54	Sequence control of position is finished. When a time value of 0 is encountered the sequence control is terminated and <Mode> changed to this 54.
55	Sequence control of speed by ramp control. The configured <RampAccMax> and <RampDecMax> are used to set acceleration.
56	Sequence control of speed by interpolation. Acceleration is set by the difference in speed between the current sequence entry and the next one, together with the sequence time value.
59	Sequence control of speed is finished.

The following table summarizes the available registers for sequence control:

Settings for SeqControl:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
												Repeat	Time Scale		Delay

Delay
When delay mode = 0
The countdown of the time set in the current sequence <SeqTime[]> starts immediately. When the countdown is done the next sequence starts, even if the set position or speed set in <SeqTarget[]> isn't reached.
When delay mode = 1, the timer starts when the set position or speed is reached.

Time scale
Sets the time scale of the timer.
Bit 1 and 2 = 0, time in milliseconds
Bit 1 = 1, bit 2 = 0, time in seconds
Bit 1 = 0 and bit 2 = 1, time in minutes.

Repeat
When Repeat = 0, the sequence will end when encountering <SeqTime[]> = 0,
When Repeat = 1 the sequence will continue regardless of <SeqTime[]> = 0.

The following table summarizes the available registers for sequence control:

Name	Type	Nr	Description
SeqControl	uns16	500	.

Bits	Name	Description
0	Delay mode	When delay mode = 0, timer set by <SeqTime[]> starts immediately. When <SeqTime[]> is up, the function moves to the next sequence entry, even if the set position or speed isn't reached. When delay mode = 1, the timer starts when the set position or speed is reached.
1-2	Time Scale	Sets the time scale of the timer. Bit 1 and 2 = 0, time in milliseconds Bit 1 = 1, bit 2 = 0, time in seconds Bit 1 = 0 and bit 2 = 1, time in minutes.
3	Repeat	When Repeat = 0, the sequence will end when encountering <SeqTime[]> = 0, When Repeat = 1 the sequence will continue regardless of <SeqTime[]> = 0.

Name	Type	Nr	Description
SeqIndex	uns16	501	Current index into the table of positions and time values. Can have values from 0 to 15. During sequence control this register automatically increments for each processed table entry.
SeqTarget[15]	int32	510-541	Target values of the table. Position: Unit is 1/4096 revolution Speed: Positions/second / 16
SeqTime[15]	uns16	570-585	Time values of the table. Unit is milliseconds, seconds or minutes.

4.6 Homing

In many applications the position control is in absolute terms. This requires the system to obtain a position reference at startup. This procedure is commonly termed ‘homing’ or ‘referencing’ and often operates by slowly moving the motor in one direction until a home switch is engaged. When the switch is operated the motor position is reset to some known value. There are a lot of different schemes for the homing sequence though, and therefore a flexible 4-step homing sequence is supported by the Simplex Motion motor units.
The general homing speed and acceleration is set by the <HomeSpeed> and <HomeAcc> registers.
Each of the sequence steps are configured by a 16-bit <HomeSequence> register.

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Relative speed				Filter				Pola-rity	Direc-tion	Condition source					

The condition is defined by the ‘Condition source’ according to:

Value	Description
0	None, this step is disabled
1	Torque. When Polarity is set to 1, this condition becomes true if motor torque is above HomingTorque register value.
2	Digital input IN1
3	Digital input IN2
4	Digital input IN3
5	Digital input IN4
6	Digital input IN5
7	Digital input IN6
8	Digital input IN7
9	Digital input IN8
10	The sequence runs the motor for the time specified by the ‘Filter’ bits

The ‘Direction’ bit sets the direction of the homing.

Direction	Description
0	Positive direction
1	Negative direction

The ‘Polarity’ bit decides if the condition is met when then input is high or low, or torque is above or below <HomingTorque>.

Polarity	Description
0	Condition met when: -Input is low -Torque below HomingTorque
1	Condition met when: -Input is high -Torque above HomingTorque

There is a filtering feature as well if the condition input is noisy. This works by requiring the condition to be true a number of times in a sequence. The ‘Filter’ entry can be set to 0-15, which selects a number of regulator cycles according to the table below.

Setting	Cycles	Running Time
0	No filter	
1	2	0.5ms
2	4	2ms
3	8	4ms
4	16	8ms
5	32	16ms
6	64	32ms
7	128	64ms
8	256	128ms
9	512	256ms
10	1024	1.02s
11	2048	2.05s
12	4096	4.10s

13	8192	8.19s
14	16384	16.4s
15	32768	32.8s

To allow different speeds for the homing sequence steps there is also a 4-bit 'Relative speed' entry. The value 0-15 is interpreted as relative speed 0-100% of the <HomeSpeed> register value.

When all of the 4 steps have been completed (those that are not used should be set to 0) the <MotorPosition> register is reset to the value in the <HomeOffset32> register. The difference between the actual <MotorPosition> value at this instant, and the <HomeOffset> value, is stored in the <HomeChange> register. This value makes it easy to check the precision of the homing sequence by performing it repeatedly and studying the <HomeChange> register contents.
The <HomeDoneMode> register is used to change the <Mode> register when the homing sequence has completed. This is useful for example to jump right into position regulation mode after the homing sequence is finished. For standalone operation it might be useful to store the homing mode in the non volatile memory to make the system automatically perform homing at power on, and then entering the position regulation mode when homing is completed.

Related to homing are 'Limit switches', that are typically used to turn off the driver when the position approaches a mechanical stop to avoid damage. Support for limit switches works by using any of the digital inputs, and then specifying this input to be monitored in the status register. This enables 'Shutdown' or 'Quickstop' modes to be automatically asserted from activating these inputs. Read more in section 0 about this.
It may also be possible to avoid end switches completely in an application by carefully setting the maximum motor torque and assuring that there are mechanical stops that can withstand this torque. The homing sequence can use torque sensing to detect the reference position, and the status bit 'Torque' can be used to automatically disable the motor ('Shutdown' or 'Quickstop' modes).

The following table summarizes the available registers for homing control:

Name	Type	Nr	
HomeSequence[4]	uns16	480-483	Sequence definition for homing sequence. 4 individual sequence steps. The homing features are used to find a position reference at system startup.
HomeOffset16	int16	490	New position value the homing position should have in relation to the 0-position. Only used for legacy reasons. Use <HomeOffset32> instead
HomeSpeed	uns16	491	Reference speed to use for homing. Unit is positions/second / 16.
HomeAcc	uns16	492	Homing acceleration. Unit is positions/second^2 / 256.
HomeTorque	uns16	493	Torque limit to use by hard stop homing. Unit is mNm.
HomeDoneMode	uns16	494	Mode to switch to when homing sequence is finished. It is then written to register 400.
HomeChange	int16	495	The amount of position change after a completed homing. Useful for debugging and basically shows how much repetitive homing deviates.
HomeOffset32	Int32	496	New position value the homing position should have in relation to the 0-position. Allows the 0 position to be placed at an offset from the homing position.

4.7 Events

⚠ WARNING

EVENTS SAFETY
Improper use of events could disable other function such as "Shutdown" and "Quickstop" etc. that could lead to a hazardous situation. Take great care when using events and don't use them to disable other safety functions.
Failure to follow these instructions can result in death or serious injury

To make stand alone operation of the unit possible, a feature called event handling is available. It solves the task of letting digital inputs, for example connected to pushbuttons, affect registers such as increasing the speed, stopping the motor etc. Or setting digital outputs based on register contents such as 'motor position is larger than xxx'.
There are 20 separate and independent events available. Each event is evaluated each regulator cycle at 2kHz.

The configuration of Events is easily done using Simplex Motion Tool, available from the Simplex Motion website., www.simplexmotion.com.

Events are based on trigger conditions that act on a selected register. When a trigger is activated, another register manipulation is executed. By manipulating registers, it is possible to change the motor operation, set a digital output, or control any other aspect of the motor unit.

4.7.1 Event trigger

A trigger condition is met when a register content together with an operator and a data value produces a non zero result. Any register can be selected by entering the register number in the <EventTrgReg> register. There are 16 operators to choose from, and the selection is done by setting the 4 bits at bit positions 0..3 in the <EventControl> register. The data value used is entered in the <EventTrgData> register. The trigger value is calculated as follows, and the trigger becomes active when this value is nonzero.

Trigger value = <Register> OPERATOR DataValue

The following operators are available:

Value:	Operator:	
0		Always true
1	=	Equal
2	!=	Not equal
3	<	Less than
4	>	Greater than
5	or	Bitwise or
6	nor	Bitwise not or
7	and	Bitwise and
8	nand	Bitwise not and
9	xor	Bitwise exclusive or
10	nxor	Bitwise not exclusive or
11	+	Add
12	-	Subtract
13	*	Multiply
14	/	Divide
15	Value	Takes data value directly

The trigger can also be filtered to increase rejection to noise (for example pushbutton debouncing) or to create a time delay. The filter will require the trigger evaluation to be active a certain number of times in a row before it is interpreted as activated.
Together with the 'repeat' trigger type it also allows the event to be executed at a controlled repetition rate when the trigger condition is continuously true. This can for example be used to repeatedly increase the position of the motor when a pushbutton is being held pressed for a long time.
The filter is configured by the 4 bits at bit positions 4..7 in the <EventControl> register according to:

Setting	Evaluations	Time delay
0	No filter	

1	2	1.0ms
2	4	2ms
3	8	4ms
4	16	8ms
5	32	16ms
6	64	32ms
7	128	64ms
8	256	128ms
9	512	256ms
10	1024	512ms
11	2048	1.02s
12	4096	2.05s
13	8192	4.10s
14	16384	8.19s
15	32768	16.4s

The trigger can also have different types of behavior to further expand the flexibility. See the following table for the 3 types of triggers available. The type is configured by setting the bit positions 8..11 of the <EventControl> register.

Setting	Trigger type	Description
0	Active	Event is performed each time the filtered trigger condition is true.
1	Edge	Event is only performed the first time the filtered trigger condition becomes true. The trigger condition has to become deactivated again before next trigger can occur.
2	Repeat	Event is performed repeatedly while the trigger condition is true, but the filter is reset each time so that the filter creates a time delay between event executions.

4.7.2 Event execution

When a trigger condition is determined true, the event is executed. This is done by taking the contents from a source register, and together with an operator and a data value, create a new value that is then written to a destination register. This makes many register manipulations possible, such as setting a constant value in the register, moving one register content to another register, setting one bit in a register, increasing the value in a register etc.

The source register is specified by entering the register number in the <EventSrcReg> register. The operator is selected by the bit positions 12..15 in the <EventControl> register. The data vale is taken from the <EventSrcData> register. The final value is written back to the register specified by the <EventDstReg> register.
The value is calculated by:

Value = <Register> OPERATOR DataValue

The available operators are (same as for triggering):

Value:	Operator:	
0		Always true
1	=	Equal
2	!=	Not equal
3	<	Less than
4	>	Greater than
5	or	Bitwise or
6	nor	Bitwise not or
7	and	Bitwise and
8	nand	Bitwise not and
9	xor	Bitwise exclusive or
10	nxor	Bitwise not exclusive or
11	+	Add
12	-	Subtract
13	*	Multiply
14	/	Divide
15	Value	Takes data value directly

To summarize the <EventControl> register contents:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Data operation				Trigger type				Trigger filter				Trigger operation			

The event is disabled by setting the EventControl register to 0. Setting Trigger operation to 0 makes the event executed for every regulator cycle.

Summary of registers for event handling:

Name	Type	Nr	Description
EventControl[20]	uns16	680-699	Control register for event. **Bits** / **Description**: 0..3: Trigger operation Used to determine if trigger condition is met. 4..7: Trigger filter Allows filtering of trigger condition. Values 0..15 corresponds to filter delay times of 1, 2, 4, 8, ... 32768 regulator periods. 8..9: Trigger type 0 = Active, 1 = Edge, 2 = Repeat. 10..13: Data operation Used to manipulate register when event is executed.
EventTrgReg[20]	uns16	700-719	Trigger register number.
EventTrgData[20]	uns16	720-739	Trigger data value. 16-bit value to use with trigger register and operator.
EventSrcReg[20]	uns16	740-759	Source register number.
EventSrcData[20]	uns16	760-779	Source data value. 16-bit value to use with source register and operator.
EventDstReg[20]	uns16	780-799	Destination register number to write event execution result to.

4.8 Recorder

To facilitate measuring of the unit behavior and performance there is an internal recorder to record parameters over time. It is capable of 4 channels, 500 measurements and up to 2kHz recording speed. Each channel is 16 bits wide, so capturing full 32 bit registers is not possible.

The <RecState> register determines the recorder state and can be both read and written. Both continuous recording and one single recording of 500 values can be started. It is also possible to set a trigger condition for recording. In that case the recorder is first run continuously while waiting for the trigger condition to be met. When the trigger occurs, it continues for a number of samples equal to 500 – <RecPreceding> register. The <RecPreceding> register makes it possible to inspect what happens just prior to the trigger. Since the recorder collects data continuously in a circular buffer, a triggered recording has its first data at an offset into the data buffers, specified by the register <RecOffset>.

The speed of the recorder is set in number of regulator cycles to wait between recordings. Values 0..65535 are valid, where 0 is the fastest possible at 2kHz, taking all 500 measurements in 0.25 seconds.

The trigger condition is set by the <RecTrigger> register. The register is used as a mask for the status bits (see section 0). When a bit in the status register is set, and the corresponding bit in this register is set, the trigger condition is true. This allows triggering from one or more of the bits in the status register. For more advanced triggers, for example triggering on a single specific error code, the event handling feature can be used to set the available user configurable status bits.

Summary of registers for the recorder:

Name	Type	Nr	Description
RecState	uns16	900	State of the recorder. The recorder is used to store measurements in a rapid pace for debugging and inspection of dynamic behavior. There is space for 1000 measurements of 4 channels, each being 16 bits wide.

Value	Name	Description
0	Idle	Recorder in idle, not used.
1	Continuous	Recording continuously
2	Single	One complete recording of 1000 values done
3	Trigger	Trigger enabled; Recording starts when trigger condition is met.

Name	Type	Nr	Description
RecTrigger	uns16	901	Trigger word. This word is used as a mask with the status register. When an active status bit corresponding to an active RecTrigger bit appears the trigger condition is met.
RecPeriod	uns16	902	Sets the recording speed as number of regulator cycles between recordings. Setting this value to 0 provides the fastest possible recording speed, taking all 500 measurements in about 0.25s.
RecPreceding	uns16	903	Sets the number of samples to appear before trigger. This feature makes it possible to measure just prior to trigger condition.
RecOffset	uns16	904	Indicates the start of recorded data. Since the data area is stored in a circular buffer that runs continuously, the first data point is not always in the first memory position. Instead, the beginning of recording is at the RecOffset position.
RecRegister[4]	uns16	905-908	Register numbers for the 4 channels to record.
RecData1[500]	int16	1000-1499	Data for recording channel 1. Data can be uns16 or int16 depending on the source register. 500 values in consecutive register addresses.
RecData2[500]	int16	2000-2499	Data for recording channel 1. Data can be uns16 or int16 depending on the source register. 500 values in consecutive register addresses.
RecData3[500]	int16	3000-3499	Data for recording channel 1. Data can be uns16 or int16 depending on the source register. 500 values in consecutive register addresses.
RecData4[500]	int16	4000-4499	Data for recording channel 1. Data can be uns16 or int16 depending on the source register. 500 values in consecutive register addresses.

4.9 External inputs and outputs

There are a total of 8 external connections available for user applications. Some of these connections can have multiple uses. The following functions are available:

- QEA / QEB: for the quadrature encoder interface (or step motor emulation interface).
- RS485A / RS485B: for the Modbus or CAN interface.
- OUT1-4: Digital outputs. Open drain output with transistor that pulls output to ground.
- IN1-4: Digital or Analog Inputs 0..+3.3V or 0..+5V depending on model. (Shared connections with OUT1..4).
- IN5..8: Digital inputs 0..+3.3V or 0..+5V depending on model. (Shared connections with QEA/QEB and RS485A/RS485B).

4.9.1 Inputs

The digital inputs fall into two groups of 4 inputs.
The first group IN1-4 are also usable as analog inputs or open collector digital outputs. They have a configurable threshold level in the range 0.. +3.3V or +5V by use of the register <InputThreshold>. A value of 0 equals 0V, and 65535 equals +3.3V or +5V. The inputs can withstand up to +30V. There is a configurable pullup/down resistor of 10kOhm (see more at the description of the digital outputs). By using the pullup resistor option, it is easy to connect pushbuttons by connecting them to the input and GND.

The second group IN4-8 shares its functionality with other features. They have typical TTL logic levels, requiring the input voltage to be <0.7V for a low level and >2.4V for a high level. There is a pullup resistor to +3.3V to set the inputs high when nothing is connected. These inputs can withstand up to 8V continuously. This makes pushbutton connections simple. The IN4-8 inputs have a faster time response than the IN1-4 inputs.
All 8 inputs are accessed as 8 bits from the same register <Input> according to:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
								IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1

The active level for the inputs can be configured by the register <InputPolarity>. By setting a bit to 1 the polarity of the respective input is inverted. Setting a bit to 0 keeps the same polarity as seen by the hardware.

The analog inputs from IN1-4 are converted from 0..+3.3V or 0..+5V into 16 bit values 0..65535. They are updated each regulator cycle and filtered to limit noise. They are available in the registers <Analog>. The input impedance is 300kOhm, which requires the voltage source to have a significantly lower impedance than this. Recommended source impedance is < 10kOhm.
The analog conversions have originally 12 bits of precision but filtering and conversion to 16bits allow somewhat higher resolution.

The registers for external inputs:

Name	Type	Nr	Description
InputPolarity	Uns16	140	The 8 lower bits control input polarity on the inputs IN1-IN7. When set to 0 the corresponding input is active high, while it is active low if set to 1.
InputThreshold	uns16	141	Threshold level for low/high for the inputs IN1-4. The 16bit value represents the range 0..+3.3V or 0..+5V.
Input	uns16	145	8 bits hold states for digital inputs, IN1 in least significant bit. 1 = high level.
Analog[4]	uns16	170-173	Values from analog conversions. All values are full 16 bits that represent 0..+3.3V or 0..+5V inputs.

Nr	Description
170	IN1
171	IN2
172	IN3
173	IN4

4.9.2 Outputs

There are 4 identical digital outputs OUT1..4. These outputs are shared with digital or analog inputs. They are configured as open collector outputs, sinking up to 1A current to ground. They can withstand a voltage of between +5V to +30V depending on the series. There are resistors of 10kOhm that can be enabled as pull up resistors to +3.3V, pull down resistors to GND or disabled. For fast switching signals (such as high speed PWM) it is recommended to use an external pull up resistor of lower value, for example 1kOhm.

The actual state of the outputs can be controlled in several ways to allow advanced control such as pulses and PWM (pulse width modulation) output.
The configuration of each output is done by its respective <OutputControl> register:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
								Output mode				Pull up/down		Polarity	

Output polarity setting:

Value	Description
0	Active low input. This means that the transistor is turned on when the output is active, allowing current to flow from the output to ground. The transistor is turned off when the output is deactivated.
1	Active high input. The transistor is turned off when the output is active, and a pull up resistor is needed to set the output voltage high. The transistor is turned on and conducts current to ground when the output is deactivated.

Pull up/down resistor setting:

Value	Description
0	None. The pull up/down resistor is disabled.
1	10kOhm pull up resistor to +3.3V enabled.
2	10kOhm pull down resistor to GND enabled.

And the mode setting is defined by:

Value	Name	Description
0	Digital	Simple digital output. When the <Output#> register is non zero the output is set active. When the <Output> register is zero, it is deactivated. (Note: Only available on SC, SM and SH series. For simple output, please use mode 8)
1	PulseShort	Single short output pulse Every time the <Output> register is written, a pulse will be generated. The length of the pulse is controlled by the value of the <Output> register. The time is controlled in units of 1us, so pulses from 1us..65ms are possible.
2	PulseLong	Single long output pulse Every time the <Output> register is written, a pulse will be generated. The length of the pulse is controlled by the value of the <Output> register. The time is controlled in units of 1ms, so pulses from 1ms..65s are possible.
3	Pwm16	The output uses pulse width modulation with 16 bits of resolution. PWM frequency is 0.92kHz.
4	Pwm14	The output uses pulse width modulation with 14 bits of resolution. PWM frequency is 3.7kHz. The Output value still uses all 16 bits.
5	Pwm12	The output uses pulse width modulation with 12 bits of resolution. PWM frequency is 15kHz. The Output value still uses all 16 bits.
6	Pwm10	The output uses pulse width modulation with 10 bits of resolution. PWM frequency is 59kHz. The Output value still uses all 16 bits.
7	RcServo	Generates a pulse length 1.0..2.0ms long every 20ms, which is appropriate to feed to an RC hobby servo device. Output value of 0 produces 1.0ms pulses, while the maximum value of 65535 produces a pulse width of 2.0ms. Set Polarity = 1 for compatibility with RC servos.
8	Output	Simple digital output. When the <Output#> register is non zero the output is set active. When the <Output> register is zero, it is deactivated.

When using PWM or pulses to switch heavy inductive loads, such as solenoids or motors, it is important to cater for recirculating currents in the load. A switching diode rated at 1A or more should be connected across the load, with the anode to the digital output and the cathode to the power supply used for the load.
The PWM output mode can be used for controlling small motors, clutches, solenoids, lamps etc.

Registers for control of digital outputs:

Name	Type	Nr	Description
OutputControl[4]	uns16	150-153	This register controls the mode of a digital output, allowing simple, pulse, PWM or RC servo pulse output. It also configures the pull up/down resistor and output polarity.
Output[4]	uns16	160-163	The output value. This value is interpreted differently depending on the output modes set in the respective <OutputControl> register.

4.9.3 Encoder

The encoder interface is a versatile interface that supports several different input and output modes. The following table describes the different modes:

Mode	Description
Quadrature encoder	The quadrature encoder feature is available as an extra input. It uses 90-degree phase shifted signals to sense both movement and direction. The ENCA and ENCB signals are TTL logic +5V inputs. Pulse frequencies up to 2.2MHz are supported, depending on filter settings. The count rate is 4 times the pulse rate from the encoder, as the interface counts all the phases. So, a 500PPR encoder will produce 2000 counts per revolution. The Encoder value can be used as a target for the PID controller, which can be useful to let this motor drive unit track another mechanical motion, such as another motor. The target value scaling feature allows electronic gearing in such applications.
Step/Direction	The encoder interface can also be used to implement a step/direction interface. This type of interface, with a logic direction select input signal to select direction of rotation, and a step pulse input, is standard for step motor drivers. The encoder count is incremented or decremented one unit for each complete input pulse. By implementing this type of interface, it is very easy to replace stepper motors in many applications, with significant improvements in performance.
RC servo pulse	This mode allows control of the motor from a RC (Hobby radio control) system. These systems utilize a pulse length control method. The input pulses will result in a signed 16bit value stored in the <Encoder> register. The value will be scaled according to:

Pulse length	Value
1.0ms	-32768 (min)
1.5ms	0
2.0ms	32767 (max)

Mode	Description
	The input signal should be connected to the ENCA input. There is also a timeout feature that sets the value to 0 if no pulses appear for 100ms.
Encoder output	Another option is to use the ENCA/B signals as quadrature encoder outputs. Pulses will be generated to match the motor movement as represented by the register <MotorPosition>. This feature is very useful for synchronizing two motors, where the master motor has ENCA/B setup as encoder outputs, and the slave motor uses ENCA/B as encoder inputs and <TargetSelect> setup to use the encoder value as the motor position target. Note: Encoder output is available on all models from firmware revision 01.30, but only on SH/SM series on earlier revisions. On SC-series motors the encoder output shares resources with pulse/PWM outputs on INOUT2/3, so limitations apply when it comes to simultaneous use of encoder output and Pulse/PWM output modes. **NOTE!** Using the encoder output for motor synchronization with long cables can be problematic because of interference and/or ground potential differences. There is an accessory product available to solve the problems with differential encoder signaling. Please see www.simplexmotion.com for more information.

The <EncoderControl> register configures the encoder interface:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
						Pullup	Dir	Encoder filter				Encoder mode			

Parameter	
Encoder mode	0 = Encoder interface disabled 1 = Standard Quadrature encoder input mode 2 = Step motor emulation, ENCA = Step, ENCB = Direction. 3 = RC servo pulse input on ENCA pin. 8 = Quadrature encoder output mode (Applicable to motors from SM-Series and SH-Series. Not applicable to SC-Series)
Encoder filter	Allows digital filtering of the ENCA/ENCB inputs. It is a common problem with electrical noise on encoder signals, and filtering minimizes these issues. But filtering also limits the maximum frequency on the signals. Values are 0..7, where 0 is minimum filtering. Default value is 4.

Value	Max pulse frequency
0	10 MHz
1	5 MHz
2	2.5 MHz
3	1.25 MHz
4	625 kHz
5	312 kHz
6	156 kHz
7	78 kHz

Direction	Direction control. 0 = ENCA leading ENCB equals positive direction 1 = ENCB leading ENCA equals positive direction
Pullup	0 = Pull down resistor connected to the input (default and recommended) 1 = Pull up resistor connected to the input (allows operation with open collector drivers)

It is possible to both read and write to the <Encoder> register, and the value is maintained when updating the <EncoderControl> register.

The registers used for the encoder:

Name	Type	Nr	Description
EncoderControl	uns16	180	Controls function of quadrature encoder inputs.
Encoder	int32	184/185	Value from quadrature encoder interface. Counts 4 * pulse frequency from encoder. In case Encoder mode = 2 (step motor emulation) this register holds the counter value from the step/dir interface instead.

Indicator LEDs

There are indicator lights on the unit showing the current status of the device according to:

Indicator	Status
Steady Green	Power is on, motor is off or standstill.
Blinking Green	Motor is moving or torque is applied. The blink frequency increases with motor speed. The color will shift to red when the torque increases to high values.
Short Red blink	can be several reasons: o The mode register is changed. o The non volatile memory is written
Red light with green blinks	Error state. The number of consecutive yellow blinks indicate which status bit caused the error. This table shows the error cause:

Nr of blinks	Description
1	Internal fault in the unit
2	USB, Modbus or CAN communication error
3	Motor current is too high
4	Supply voltage too low or too high
5	Temperature is too high
6	Motor torque is above set threshold
7	Shaft is locked (power applied but not moving)
8	The regulator has a large error, <RegErrorMax> > <RegError>
9	Motor is rotating, speed > 0.1rps
10	Motor is rotating in reverse direction
11	Target reached when ramping position control
12	For future use
13	Digital input, specified by the <StatusInputs> register.
14	Digital input, specified by the <StatusInputs> register.
15	For user application, set by event handler.
16	For user application, set by event handler.

Note that several of these cases are not typical errors, but all status bits can be configured to set the motor in error state.

Green and Red alternating blink	Motor is in Firmware mode. To exit Firmware mode without upgrading the firmware requires a power cycle.

5 Additional features

This section describes additional features and functions.

5.1 Cogging compensation

The type of motor used in Simplex Motion products, the outer rotor permanent magnet synchronous machine, has considerable cogging torque. This is the somewhat un-regular torque felt when rotating the motor shaft of a non- powered motor unit. The cogging torque is a result of the varying magnetic reluctance across different rotor angles. As this torque is repeatable it is possible to measure it and compensate for it during motor operation. Doing this will enhance speed stability at low speeds and cause less vibrations.

To use this feature a cogging calibration is needed. This calibration is done one time only and stored in the non volatile memory. Once it has been done it is then used in all motor operation modes and no further action is needed. This calibration is currently not performed on delivered motors as standard.

5.1.1 Cogging calibration

The calibration is performed by the motor slowly rotating one complete turn. It is important that no mechanical parts are attached to the motor shaft, and that the motor is not loaded in any way when performing the calibration.
The calibration is started by setting the 400 <Mode> register to 110. The <Mode> value will automatically change to 0 when the calibration is finished.
To remove the calibration and disable the cogging compensation function, set register 400 <Mode> to 111.

5.2 Motor heating

When the Simplex Motion motors are utilized in environments with very low temperatures extra heating may be needed to maintain operational temperature for the motor itself and surrounding parts. A feature has been implemented that utilizes the motor winding for thermal heating. Power from the electrical power supply is converted to heat by allowing a controlled current through the motor windings.

Heating operation is only possible in the active modes where the motor performs position (Mode 21) or speed control (mode 33). When the motor is turned off (Mode 0) or in error state (Mode 4) the heating is disabled.

The settings define a power level to use for heating, and a target temperature for the heating. The set power is applied to the motor winding until the temperature gets close to the target temperature. The power is scaled down when getting closer to the target temperature to avoid overshoot and allow continuous low power heating to maintain temperature. The scale down range is currently 5 degrees.
The measured temperature on the motor electronics is used for the temperature regulation. But the estimated motor winding temperature is also taken into account, and the heating power is decreased if the motor winding temperature gets close to the maximum allowed temperature.
There is currently no secondary protection feature to guard against single faults of the heating functionality.

Register 105 <Heating> controls the motor heating feature. The default value is 0, disabling this feature.

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
								Temperature				Power			

The heating power level is set by the lowest 4 bits:

Value	Description
0	Heating functionality disabled
1	Low power level (half of nominal)
2	Nominal power level. The actual power depends on motor model:
3	High power level (double of nominal)

Motor model	Power
SC010x, SE010x	5W
SC020x, SE020x	10W
SC040x, SE040x, SH100x, SM100x	20W
SH200x	40W

Temperature is set according to:

Value	Temperature setting
0	-40°C
1	-30°C
2	-20°C
3	-10°C
4	0°C
5	10°C
6	20°C
7	30°C
8	40°C

A typical setting to heat the motor to 0°C with nominal power would be 0x0042 hexadecimal, or 66 decimal value.

5.3 Brake

The motor can be used to control an external brake. The brake can be directly connected between motor power supply and one of the I/Os. The brake functionality is controlled by the <Brake> and <Brake timing> registers.

5.3.1 Connection

The SH100B/C, SH200B and selected models of SE020A and SE040A, have integrated electronics for direct connection of the brake to the motor. See the Technical Data Sheet for respective model for more information.

5.3.2 Settings

The brake settings are divided into two registers.
The <BrakeSetting> register:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Enable	Active	Active low	Control mode		Actuation force			Hold force						Brake I/O	

Brake I/O:

Bits	Output connected to brake
0..1	0 = OUT1 1 = OUT2 2 = OUT3 3 = OUT4

The brake can be connected to the I/O 1-4.
On selected models, electronics for using the brake is integrated and connected to a preset I/O. See Technical Data Sheet for more info.

Hold force

Bits	Hold force (% of Actuation force)
5..7	0 = 12.5% 1 = 25% 2 = 37.5% 3 = 50% 4 = 62.5% 5 = 75% 6 = 87.5% 7 = 100%

Actuation force

Bits	Actuation force
8..10	0 = 12.5% 1 = 25% 2 = 37.5% 3 = 50%

	4 = 62.5% 5 = 75% 6 = 87.5% 7 = 100%

Control Mode

Bits	Control mode
11..12	0 = On Mode 1 = On Target

Active low

Bits	Active low
13	0 = Used if the brake is braking when it is not activated 1 = Used if the brake is braking when it is activated

Active (Status)

Bits	Active
14	0 = Brake inactive 1 = Brake active

Enable

Bits	Brake function
15	0 = Brake function disabled 1 = Brake function enabled

The <BrakeTiming> register:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Release time								Engage time							

Engage time

Bits	Description
0..7	0 -255 ms

Release time

Bits	Description
8..15	0-255 ms

Example:
To release a brake with the brake connected to OUT1:
Brake I/O = 1
Hold force = Lower force (voltage) than the Actuation force. In this case 50% is used, which gives 3 and 0111 in binary.
Actuation force = The force (voltage) used to release the brake. In this case it is assumed that the motor and the brake have the same rated voltage, so 7 is used, which equals 1111 in binary
Control Mode = 1, Brake is enabled when target is reached.
Active low = 0 (the brake is engaged when it is not activated)
Active = 0 (Status bit that shows if the brake is active or not)
Enable = 1 (brake function is enabled.

Combine the binary digits together in the bitfields:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Enable	Active	Active low	Control mode		Actuation force			Hold force						Brake I/O	
1	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**1**

1000111101100001 equals 36705 in <BrakeSettings> register

Engage time = 50ms which equals 0011 0010

Release time = 50ms which equals 0011 0010

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Release time								Engage time							
0	0	1	1	0	0	1	0	0	0	1	1	0	0	1	0

0011 0010 0011 0010 equals 12850 in decimal, which is written to <BrakeTiming>.

5.4 Energy dump

In applications with dynamic motion control, where a mechanical load is both accelerated and decelerated, the motor operates as both a motor and generator. When the high inertia load is decelerated the motor converts mechanical energy into electrical energy, producing an electric current out from the motor. In battery supplied applications this feedback power can be used for battery charging.
But in applications supplied by a mains connected power supply there is usually nowhere this energy can be consumed, and instead the system capacitance is charged and the supply voltage increases. Potentially the supply voltage can reach levels that are higher than the components connected to the supply line can handle, and damage may occur.
The most common solution to this problem is to convert the excess electrical energy into heat by using some sort of 'braking resistor'.
The energy dumping feature is included in the overvoltage register, which allows the motor to monitor and control the sinking of energy in an external power resistor.

5.4.1 Hardware

An external hardware consisting of a large power resistor and a mosfet transistor switch that can turn on/off quickly is need for the power dump feature.
Simplex Motion offers the hardware SMdump as an accessory.

5.4.2 Control

The control of power sinking is done by the motor. The motor monitors the measured supply voltage and starts sinking power into the resistor when the supply voltage gets above the set threshold voltage. The control uses pulse width modulation (PWM) control of the power resistor at a constant frequency of 10kHz. The PWM duty cycle is controlled in a linear fashion with 0-100% across a voltage span of 5 volts. Thus, if the threshold is configured to 26V (a reasonable level for a 24V supply system) the duty cycle is linearly increased from 0 till 100% for voltages between 26 and 31V. The energy dump feature is controlled by the OverVoltage register:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Mode					Auto voltage	Voltage limit									

Voltage limit

Bits	Voltage limit
0..9	The threshold at which to activate the voltage protection feature. The unit is in 0.1V 0 = The default motor maximum voltage rating is used.

Auto voltage

Bits	Auto voltage
10	The average idle voltage +2V will be used. Not implemented yet.

Mode
Determines what should happen the voltage limit is reached.

Bits	Mode
12..15	0 = Turn off motor 1 = Engage external brake resistor on OUT1 2 = Engage external brake resistor on OUT2 3 = Engage external brake resistor on OUT3 4 = Engage external brake resistor on OUT4

Example:
To activate an external resistor circuit connected to OUT1 at 26V, the following data should be entered:

- Voltage limit: 26V converted to the unit of the voltage limit field is 260. 260 is 01 0000 0100 in 10 bit binary (bits 0..9)
- Mode = 1 which is 0001 in 4 bit binary (bits 12..15)

Combine the binary digits together in the bitfields:

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Mode					Auto voltage	Voltage limit									
0	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**

0001 0001 0000 0100 equals 4356 in decimal, which is written to <110>.

6 Protection and error handling

There are a number of protection features to minimize the risks of damaging the unit while still allowing full use of the performance. The main features are:

- Hardware overcurrent protection on motor current. This error can not be masked out, it will always trip the driver and cause shutdown. This protection can be compared with a fuse – It should not be activated by normal operation.
- Torque limit. The motor output torque is always limited to a user settable value. This limit should be set according to the application.
- Over/under voltage. The hardware includes a protection diode that conducts current when the input supply voltage is above +60V to protect the circuitry from damage. If large amounts of currents are supplied the protection diode will be damaged. This can be the case when braking large inertia loads, as all the energy is then output to the power supply, raising its voltage level. This error can not be masked out, it will always trip the driver and cause shutdown. This protection can be compared with a fuse – It should not be activated by normal operation. The present supply voltage is continuously measured and available in the <Supply> register.
- Temperature. The electronics include a temperature sensor, and by use of a thermal motor model the motor winding temperature is estimated. Both these temperatures are available through registers. This error can not be masked out, it will always trip the driver and cause shutdown. This protection can be compared with a fuse – It should not be activated by normal operation.
- Locked shaft. If no movement on the motor shaft is detected even though it is fed with significant current, a status bit is set. This could indicate a serious fault and can trip the driver if requested.
- Regulator error. The register <RegError> continuously shows the difference between actual and target values for the regulator. If this vale exceeds the value in the <RegErrorMax> register a status bit is set. This can trip the driver as well.

Status information from the unit is available through the <Status> register. Some bits indicate errors while others are more of informational use. These status bits are only active as long as the error cause is active. To ensure that no status events are missed there is also a latched version of the status register, <StatusLatched>. This register keeps status bits active until they are cleared by the user.
The <Status> register also holds the current status of two inputs. These inputs are selected from the available digital inputs by use of register <StatusInputs>. It is also possible to filter these inputs to suppress noise. This feature is useful for implementing driver shutdown from limit switches to avoid mechanical damage in a setup with limited travel.

To enable stopping of the motor driver upon errors, what is frequently termed 'driver trip', there are two mask registers that selects which status bits to monitor. If the same bit position in this mask register and the <Status> register is active at the same time the system enters the failure mode. There are two such modes:

Mode	Status mask register	Description
Quickstop	MaskQuickstop	The motor stops in a controlled fashion, usually be braking the motor with <RampDecMax> deceleration. The <Mode> register is changed to 'QuickStop'. The indicator shows a normal stop indication – A steady green light.
Shutdown	MaskShutdown	The driver turns off the motor current immediately, and if the motor was running it will continue to spin freely to a halt. The <Mode> register is set to 'Shutdown' and the indicator will show an error state where a blinking pattern indicates the source of shutdown.

The <Mode> register needs to be updated to bring the device out of the error state. Thus by setting the mask register the user can select what type of errors should trip the driver.
If more flexibility is needed, such as driver trip on a special error code, it is possible to use the User1..2 status bits that can be set using the event handler (see 0).

To provide further detail on error causes, the unit also generates error codes. These codes are 16bit with the top 4 bits equal to the status bit number to which they belong. So for example the error codes for communication errors are in the range hex[1001-1FFF]. A complete list of error codes can be found in section 6.1.

Nr	Type	Name	Description	Range:
410	uns16	Status	Drive status. Each bit has status information according to the table below. This status word is used for several things, it can trip the driver, start recording data or enable outputs. The bits are only active while the condition is true.	0..65535

Bit	Name	Description
0	Fail	Internal error in the driver
1	Communication	Communication error
2	Current	Hardware overcurrent protection triggered.

		Input current > 40A
3	Voltage	Input voltage is too high or low Voltage < 10V or Voltage > 55V
4	Temperature	Temperature of drive is too high, motor temp > 120°C or electronics temp > 100°C.
5	Torque	Motor torque limit active
6	Locked	Shaft is locked (power applied but not moving)
7	Regulator	The regulator has a large error, <RegErrorMax> > <RegError>
8	Moving	Motor is rotating Speed > 0.1rps
9	Reverse	Motor is rotating in reverse direction
10	Target	Target reached when ramping position control
11	Reserved	For future use
12	InputA	Digital input, specified by the <StatusInputs> register.
13	InputB	Digital input, specified by the <StatusInputs> register.
14	User1	For user application, set by event handler.
15	User2	For user application, set by event handler.

411	uns16	StatusLatched	A latched version of the <Status> register. The bits in this register are set when the corresponding bits are set in the <Status> register. The register is kept set until read, after which it is reset again.	0..65535
412	uns16	StatusInputs	This register defines two digital inputs that are available in the status register as InputA and InputB. This is useful for Limit switches that should cause a driver shutdown. It is also possible to filter these inputs from noise.	

Bits	Description
0..3	Input number to use for InputA. 0 = IN1, 1 = IN2 etc
4..7	Input number to use for InputB 0 = IN1, 1 = IN2 etc
8..15	Filter value. 0 = no filtering. Increasing values causes more filtering and larger delay.

413	uns16	MaskQuickstop	This is a 16 bit mask that defines which status bits that should cause a quick stop of the motor. A one in a bit position enables that status bit source.	
414	uns16	MaskShutdown	This is a 16 bit mask that defines which status bits that should trip the driver (enter error mode). A one in a bit position enables that status bit source.	0..65535
415	uns16	Error	This register holds the latest generated error code.	0..65535
420/421	uns32	Time	Tracks time as 4096 counts per second. Wraps around after about 12 days.	0 .. 4294967295

6.1 List of error codes

These are the defined error codes that can be read from the <Error> register. The register content indicates the latest error, but the error cause may not any longer be existing when the register is read.

Error code (hexadecimal)	Description
0x0001	General internal error
0x0002	Internal software timing error
0x0003	Error in application code, not terminating.
0x1001	General communication error
0x1002	Reference to invalid register number
0x1101	Modbus parity error
0x1102	Modbus framing error
0x1103	Modbus overrun error
0x1104	Modbus checksum error
0x1105	Modbus illegal function code
0x1106	Modbus illegal diagnostics function code
0x2001	Hardware overcurrent protection triggered
0x3001	Supply voltage too low
0x3002	Supply voltage too high
0x4001	Temperature of electronics is too high
0x4002	Temperature of motor winding is too high
0x5001	Torque limiting is active
0x6001	Locked shaft condition detected
0x7001	Regulator error is large

6.2 Hardware reset of registers

If there is a need to resets all parameters to factory default settings and you are unable to reach the <Mode> registers factory reset mode, there is a way of doing a hardware reset to factory default settings. If connecting IN5 directly to IN7 at power on from power off, the motor will be set to factory default settings. In the same way you can put the motor in firmware mode by connecting IN5 with IN6.

7 Power supply considerations

The power supply used with the servomotor must be able to supply enough current for the application, not only for the continuous operation, which is typically up to 6A current, but also for the instantaneous higher output power during for example high accelerations.
For applications that use high braking deceleration rates, especially with high inertia loads, the power supply unit must also be able to sink current, since the motor then operates as a generator and outputs current to the power supply. Failure to sink this current will raise the supply voltage, potentially to damaging levels. External protection zener diodes can be used to handle short bursts of overvoltage, se table below.
The <Supply> register contains the real time supply voltage and this can be monitored during motion to verify that the voltage level is not raised too high. There is also a status bit in the <Status> register that indicates high voltage conditions. The motors also have an over voltage protection feature that simply turns off the motor and generates an error code if the supply voltage gets too high.

Motor rated voltage	Recommended external protection zener diodes rating
+12V	14-16V
+24V	25-30V
+48V	50-60V

Also note that the grounding potential at the motor unit will change if there are long cables and high currents because of cable resistance. Since the motor input/outputs share the same grounding potential as the power supply, they are affected accordingly. This issue is handled by using thicker cables and/or by using the motor ground as the central grounding point (star grounding).
There are also accessory products for isolated inputs/outputs available to handle grounding issues when communicating between motors and/or other equipment.

8 EMC

EMI issues are common for motor control installations. The switching of the motor currents creates fast transients in both voltage and current waveforms that typically spread out to surrounding equipment. One of the most problematic sources of noise radiation is usually the cabling between the controller and the motor. But this issue is of no concern here since the electronics and the motor are integrated into the same enclosure with a minimum of cabling. Generally EMI problems are nonexistent in integrated drive units. But there is one issue left even for these types of implementations. The switching motor currents propagate out on the power supply cabling as well. So there will always be some noise conducted from the device to the power supply unit, and potentially radiated from the cable. In this product that noise is suppressed by two means:

- A high order low pass filter to minimize the noise above 150kHz
- Spread spectrum techniques to spread out the noise energy across a continuous range of frequencies. This limits the energy at each individual frequency.

Simples Motion motor units will pass the requirements from the IEC/EN61800-3 standard for the industrial environment. To also pass the requirements for the residential/commercial environment an external filter will have to be used on the power supply input. This can typically be a simple LC filter consisting of a 10uH inductor in series with the positive supply rail and a 100uF low ESR capacitor across the power supply feed. Both components should be located close to the Simplex Motion unit.