MAC50
MAC95
MAC140
MAC141
Including Expansion Modules

Integrated Servo Motors
Technical Manual

JVL Industri Elektronik A/S

LB0048-06GB Revised 1.5.2005

JVL Industri Elektronik A/S
Blokken 42
DK-3460 Birkerød
Denmark
Tlf. +45 45 82 44 40
Fax. +45 45 82 55 50
e-mail: jvl@jvl.dk
Internet: http://www.jvl.dk

Contents

1 Introduction

1.1 Introduction

This Technical Manual describes advanced use of the MAC50 ... MAC141 motors.

The Manual describes the functions of the MAC-motor, its data registers and interfaces in greater detail than is necessary for normal operation of the MAC motor.

Chapter 2 describes the MacRegIO software, which provides direct access for changing and reading operational parameters and data.

Chapter 3 describes all registers containing operational parameters and data.

Chapter 4 describes the function of the MAC motor in detail.

Chapter 5 describes the MAC motor's interfaces to its surroundings, i.e. RS232 and RS422 communication interfaces, and their associated protocols.

It is recommended that section 1.2 "Sampled Systems" is read. This section explains certain terminology used in sampled systems.

1.2 Sampled Systems

In contrast to analogue regulation systems, in which analogue control signals continuously flow through regulation filters, output stages, etc., the control signals in a sampled system only appear in filters and other circuitry at discrete times.

These discrete times are called the *sampling times*. The time interval between two sampling times is called a *sampling interval*. The number of samples per second is denoted as the *sampling frequency*.

At each sampling point (each sample), relevant measurement values are read, such as:

- Phase current
- Motor position
- Reference input

Values that cannot be read directly are calculated. For example:

Velocity = Delta_P / sampling interval

where Delta_P is the difference in position from the preceding sample.

At each sample, all outputs are updated, including the three phase voltages.

The MAC motor contains 3 regulation loops with a sampling frequency of 7812 Hz for regulation of the motor's 3 phase currents, and 1 regulation loop with a sampling frequency of 521 Hz for regulation of velocity / position.

The MAC motor's regulation loops contain digital filters, corresponding to the analogue filters of an analogue regulator. These digital filters are defined by and calculated in the MAC motor using the Z-transform of their transfer function.

2 MacRegIO

MacRegIO.exe is a program for directly writing to and reading from the storage registers in the MAC motor. In addition, the program provides facilities for real-time acquisition of data in a data-buffer, for writing parameter set-ups to flash memory, and for resetting the MAC motor.

2.1 Installation

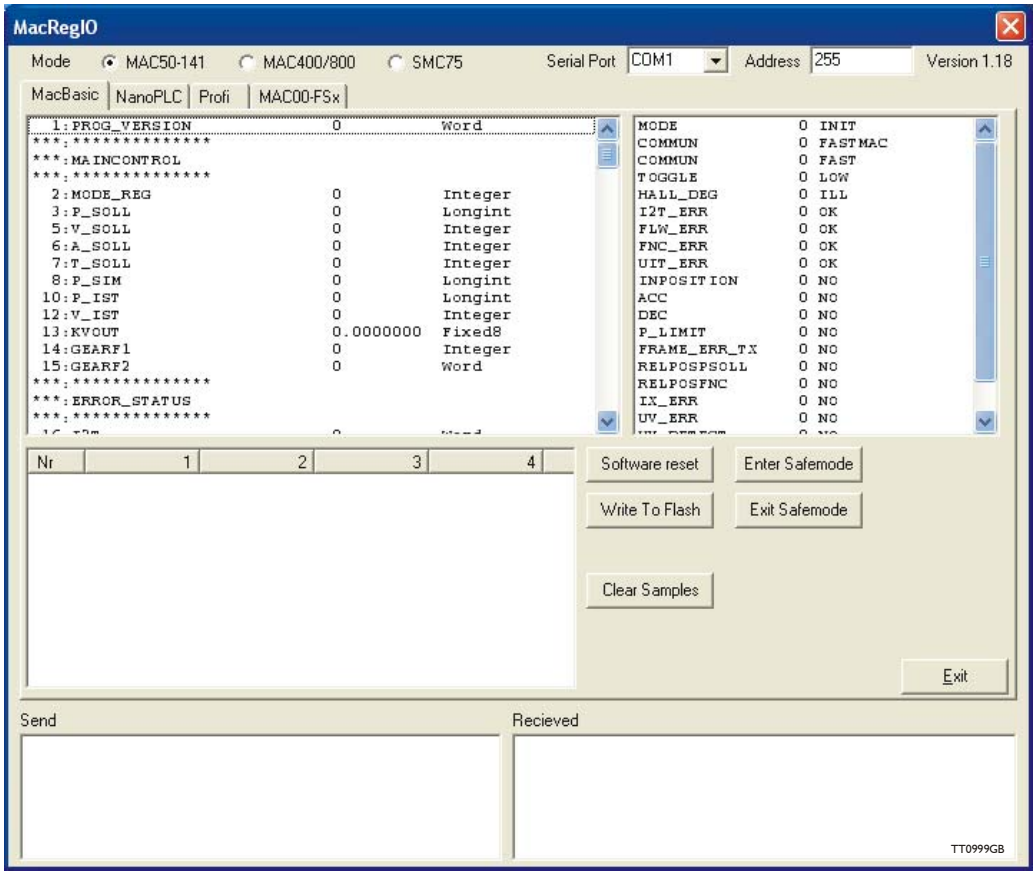
MacRegIO.exe uses two data files:

- **Bintalk.dat**: This file describes the register names and register formats for various registers.
- **Bitdefs.dat**: Certain registers contain single bits or groups of bits. This file describes the names of these bits as well as explanatory names of their status.

All three files must be installed in the same folder (directory).

2.2 Description of MacRegIO Windows

The screen of the MacRegIO software consists of 5 windows and 9 buttons, as illustrated below.



Parameter Window, (upper left):
This window displays parameter numbers, parameter names, parameter values and parameter formats. See "Description of Data Formats", page 16 for a description of the parameter formats.

Bit Window, (upper right):
This window displays the names of bits/bit groups, their numerical values and explanatory texts of the values.

Transmit Window, (lower left):
This window displays the last transmitted message, byte-by-byte, in hexadecimal format.

Receive Window, (lower right):
This window displays the last received message, byte-by-byte, in hexadecimal format.

Sample Window, (middle):
During operation, data can be gathered in real-time in a sample buffer. The number designations of the registers to be sampled are specified in registers SAMPLE1 ... SAMPLE4. Four fields are displayed at the top of the window indicating the names of the sampled registers. The register values are displayed in four columns, sample by sample. See "SAMPLE1 - Register 112", page 43.

2.2 Description of MacRegIO Windows

MacBasic Button: Always used throughout this manual.

NanoPLC Button: Never used throughout this manual.

Profi Button: Never used throughout this manual.

Software Reset Button:
This button is used to reset the MAC motor. The MAC motor can only be reset in Init or Safe Mode. See "Init Mode", page 67, "SAFE_MODE (MODE = 15)", page 78.

Write to Flash Button:
This button is used to save the setup of the MAC motor to the motor's FlashMemory. After writing the setup to Flash Memory, the MAC motor is reset. Note that writing to Flash Memory can only occur when the motor is in Safe Mode. See "SAFE_MODE (MODE = 15)", page 78.

Clear Samples Button:
This button is used to erase the contents of the Sample Window. It has no effect on the MAC motor.

Enter Safe Mode:
The MAC Motor can be equipped with various optional modules. Some of these are programmable and can control the motion of the MAC motor. If a control program is to be changed, the MAC Motor must be brought to a safe mode in which commands from the extension modules are not accepted. This button is used to set the MAC motor to Safe Mode. See "SAFE_MODE (MODE = 15)", page 78. The MAC motor will only enter Safe Mode when it is in Init Mode. On activation of this button, the MAC motor will automatically attempt to switch to Init Mode, in order to further switch to Safe Mode. See "Init Mode", page 67 and "STOP_MODE (MODE = 11)", page 75.

Exit Safe Mode:
This button is used to exit Safe Mode. The MAC motor will always automatically switch to Init Mode.

2.3 Operation

The Parameter Window is activated by clicking in the window. When the Parameter Window is active, a selected register will be highlighted.

To read the selected (highlighted) register, either the ”r” or ”R” key is used. In this way, the register contents are updated and displayed in the Parameter Window.

To write to the selected register, either the ”w” or ”W” is used. A dialogue box is then displayed and the new parameter value can be specified. **Immediately after writing to a register, the register value will be read back and updated in the parameter window. If the register is continuously updated by the MAC motor, a value other than that entered may be shown in the parameter window.**

The cursor position, i.e. selected parameter, can be changed using the arrow keys, PgUp or PgDn keys, or using the mouse.

To display the sampled register values in the Sample Window, either the ”s” or ”S” key is used. The values of the sampled registers for the first/next sample is displayed.

Note that using the”Write to Flash” button causes the MAC motor to be reset.

3 Parameter and Data Description

The following sections describe the data formats, parameter and data registers.

On delivery, the parameters are set to a default set-up. These default values are stored in ROM, (Read Only Memory). Parameter values can be changed by the user, and if the changes are required to be saved, the new parameter set-up can be written to Flash-memory. See "Description of MacRegIO Windows", page 11. This means that the new parameter set-up will be in effect each time the MAC motor is switched on.

The default values can be re-established by bringing the MAC motor to Safe Mode and writing the value 256, (100H), to the MODE_REG register. See "MODE_REG - Register 2", page 17 and "SAFE_MODE (MODE = 15)", page 78.

3.1 Description of Data Formats

The MAC motor uses 5 different data formats:

- Word
- Integer
- LongInt
- Fixed4
- Fixed8

Word: 16 bit unsigned.
Range: 0 ... 65535

Integer: 16 bit signed.
Range: -32767 ... +32767

LongInt: 32 bit signed.
Range: -2.147E9 ... +2.147E9

Fixed4: 16 bit signed fixed point.
Range: -7.999756 ... +7.999756
Unit: 1 / 4096.

Fixed8: 16 bit signed fixed point.
Range: -127.996 ... 127.996
Unit: 1 / 256

3.2 Parameter and Data Registers

The following subsections describe all of the available parameter and data registers.

Each register has its own unique register number. The following subsections of this manual are structured such that register number N is described in section 3.2.N.

Note that registers of the format LongInt take up two register numbers, corresponding to two words.

The Parameter and Data Registers can be categorised into the following groups:

- Main Control: Registers 1 ... 15. Primarily operational parameters.
- Error handling: Registers 16 ... 35. Parameters and data for error handling.
- Power on: Registers 36 ... 42. Parameters for setup and reset.
- Register mode: Registers 43 ... 88. Parameters for operation via registers.
- Filters: Registers 89 ... 111. Coefficients for various filters.
- Data acquisition: Registers 112 ... 116. Parameters for real-time data acquisition.
- Position / velocity-loop: Registers 117 ... 123 Various parameters for position- and velocity loops.
- Current-loops: Registers 124 ... 152.
- Diverse: Registers 153 ... 163.

3.2.1 PROG_VERSION - Register 1

Data format: word.

Indicates the MAC motor's program version. Can be changed by the user, but cannot be written to Flash Memory. Example: Version is read as 78 = 4Eh = version 4.14.

3.2.2 MODE_REG - Register 2

Data format: word.

Range: 0 ... 24, 256.

This register determines the mode in which the MAC motor will be operated:
Value: Operation mode:

0 Init mode. See "Init Mode", page 67 and "STOP_MODE (MODE = 11)", page 75.
1 Velocity mode, V_MODE. See "Velocity Mode", page 68.
2 Position mode, P_MODE. See "Position Mode", page 69.
3 Gear Position mode, GP_MODE. See "Gear Position Mode (GP_MODE, MODE = 3)", page 69.
4 Analog Torque mode, AT_MODE. See "Analog Torque Mode (AT_MODE, MODE = 4)", page 70.
5 Analog Velocity mode, AV_MODE. See "Analog Velocity Mode (AV_MODE, MODE = 5)", page 72.
6 Analog Velocity Gear mode, AVG_MODE. See "Analogue Velocity Gear Mode (AVG_MODE, MODE = 6)", page 72.
7 Manual current mode, MANI_MODE. See "Manual Mode (MANI_MODE, MODE = 7)", page 73.
8 Voltage test mode, TESTU_MODE. See "TESTU_MODE (MODE = 8)", page 73.
9 Acceleration test mode, TESTA_MODE. See

"TESTA_MODE (MODE = 9)", page 74.

10 Break mode, BREAK_MODE. See "BREAK_MODE (MODE = 10)", page 75.
11 Stop mode, STOP_MODE. See "STOP_MODE (MODE = 11)", page 75.
12 Home 1 mode, HOME1_MODE. See "HOME1_MODE (MODE = 12)", page 75.
13 Home 2 mode, HOME2_MODE. See "HOME2_MODE (MODE = 13)", page 76.
14 Home 3 mode, HOME3_MODE. See "HOME3_MODE (MODE = 14)", page 77.
15 SAFE_MODE. See "SAFE_MODE (MODE = 15)", page 78.
16 Analogue Velocity mode with deadband, AVZ_MODE. See “Analogue Velocity Mode with Deadband, AVZ_MODE (MODE = 16)” , page 79.
17 Velocity with Analogue Torque, VAT_MODE. See “Velocity Analogue Torque Mode, VAT_MODE (MODE = 17).” , page 79.
18 Analogue Gear, AG_MODE. See “Analogue Gear Mode, AG_MODE (MODE = 18)” , page 79.
19 Coiling, COIL_MODE. See “Coil Mode, COIL_MODE (MODE = 19)” , page 79.
20 Analogue 2 position mode, A2POS_MODE. See “Analogue 2 Position mode, A2POS_MODE (MODE = 20)” , page 82.
21 Analogue Position mode. See “Analogue Position mode, APOS_MODE (MODE = 21)” , page 82.
22 TestKI mode, TESTKI_MODE.
23 TestTQ mode, TESTTQ_MODE.
24 Gear Follow mode, GF_MODE. See “Gear Follow Mode, GF_MODE (MODE = 24)” , page 83.
256 If the MAC motor is in Safe Mode, and the value 256 is written to MODE_REG, the motor’s default parameter set-up is read. See "Parameter and Data Description", page 15 and "SAFE_MODE (MODE = 15)", page 78.

3.2.3 P_SOLL - Register 3

3.2.4

Data format: LongInt.

Range: +/-67E6.

This register indicates the absolute position to which the motor will move. See “ERR_STAT - Register 35” , page 31., bit RELPOSPSOLL.

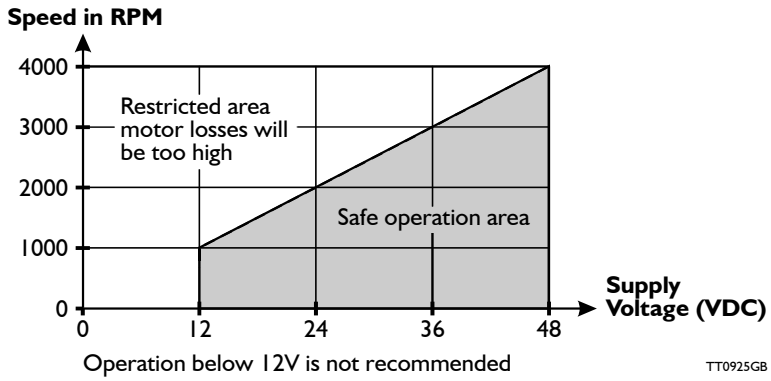
The register is used in Position mode, P_MODE, where the specified value indicates the desired position, measured in encoder counts. See the following sections: "Position Mode", page 69, "P_IST - Register 10", page 22, and "P_FNC - Register 8", page 22.

Unit: Encoder counts.

3.2.5 V_SOLL - Register 5

Data format: Integer.

Recommended range: See illustration.



This register specifies a velocity measured in 1/16 encoder counts/sample. The sampling frequency is 520.8 Hz. The number of encoder counts per revolution = 4096.

i.e. if V_SOLL is specified as 1600, the angular velocity will be:

$$\frac{1600}{16} * \frac{520.8}{4096} = 12.715 \text{ rev/sec} = 762.9 \text{ rpm}$$

V_SOLL is used differently in different operating modes:

- Velocity mode, V_MODE: V_SOLL specifies the required velocity. See "Velocity Mode", page 68.
- Position mode, P_MODE: V_SOLL specifies the maximum velocity during positioning. See "Position Mode", page 69.
- Gear Position mode, GP_MODE: V_SOLL specifies the maximum velocity during positioning.See "Gear Position Mode (GP_MODE, MODE = 3)", page 69.
- Analog Velocity mode, AV_MODE: In this mode, the motor velocity is controlled by an analogue input signal (+/- 10V). V_SOLL specifies the maximum velocity corresponding to +/- 10V. See "Analog Velocity Mode (AV_MODE, MODE = 5)", page 72.
- Analog Velocity Gear mode, AVG_MODE: In this mode, the motor velocity is controlled by signals at the pulse input, with a bias velocity that is controlled by the signal at the analogue input. V_SOLL specifies the total maximum velocity. See "Analogue Velocity Gear Mode (AVG_MODE, MODE = 6)", page 72.
- Velocity Analogue Torque mode, VAT_MODE: V_SOLL specifies the required velocity at full analogue voltage, (10 V DC). See “Velocity Analogue Torque Mode, VAT_MODE (MODE = 17).” , page 79.
- Analogue Gear mode, AG_MODE: V_SOLL specifies the max. velocity. See “Analogue Gear Mode, AG_MODE (MODE = 18)” , page 79.
- Coil mode, COIL_MODE: V_SOLL specifies the maximum velocity during positioning. See “Coil Mode, COIL_MODE (MODE = 19)” , page 79.
- Analogue 2 Position mode, A2POS_MODE: V_SOLL specifies the maximum velocity during positioning. See “Analogue 2 Position mode, A2POS_MODE (MODE = 20)” , page 82.

- Analogue Position mode, APOS_MODE: V_SOLL specifies the maximum velocity during positioning. See “Analogue Position mode, APOS_MODE (MODE = 21)” , page 82.
- Gear Follow mode, GF_MODE: V_SOLL specifies the maximum velocity, following the input pulse signal. See “Gear Follow Mode, GF_MODE (MODE = 24)” , page 83.See “Gear Position Function block” , page 62.

Unit: 0.4768 rpm.

3.2.6 A_SOLL - Register 6

Data format: Word.

Recommended range: 1 ... 1000.

This register specifies an acceleration measured in 1/16 encoder counts/sample^2. The sampling frequency is 520.8 Hz. The number of encoder counts per revolution = 4096.

i.e. if A_SOLL is specified as 160, the acceleration will be:

$$\frac{160}{16} * \frac{520.8^2}{4096} = 662.2 \text{ rev/sec}^2 = 39731 \text{ rpm/sec}$$

A_SOLL specifies the maximum acceleration/deceleration in the following modes:

- Velocity mode, V_MODE. See "Velocity Mode", page 68.
- Position mode, P_MODE. See "Position Mode", page 69.
- Gear Position mode, GP_MODE. See "Gear Position Mode (GP_MODE, MODE = 3)", page 69.
- Analog Velocity mode, AV_MODE. See "Analog Velocity Mode (AV_MODE, MODE = 5)", page 72.
- Analog Velocity Gear mode, AVG_MODE. See "Analogue Velocity Gear Mode (AVG_MODE, MODE = 6)", page 72.
- Homing mode 1 ... 3. See "HOME1_MODE (MODE = 12)", page 75, "HOME2_MODE (MODE = 13)", page 76, and "HOME3_MODE (MODE = 14)", page 77.
- Velocity Analogue Torque mode, VAT_MODE. See “Velocity Analogue Torque Mode, VAT_MODE (MODE = 17).” , page 79.
- Analogue Gear mode, AG_MODE. See “Analogue Gear Mode, AG_MODE (MODE = 18)” , page 79.
- Coil mode, COIL_MODE. See “Coil Mode, COIL_MODE (MODE = 19)” , page 79.
- Analogue 2 Position mode, A2POS_MODE.
- Analogue Position mode, APOS_MODE. See “Analogue Position mode, APOS_MODE (MODE = 21)” , page 82.
- Gear Follow mode, GF_MODE. See “Gear Follow Mode, GF_MODE (MODE = 24)” , page 83.

Unit: 248.3 rpm/sec.

3.2.7 T_SOLL - Register 7

Format: Word.

Range: 0 ... 1023

This register specifies the maximum allowable torque in the following modes:

- Velocity mode, V_MODE. See "Velocity Mode", page 68.
- Position mode, P_MODE. See "Position Mode", page 69.
- Gear Position mode, GP_MODE. See "Gear Position Mode (GP_MODE, MODE = 3)", page 69.
- Analog Velocity mode, AV_MODE. See "Analog Velocity Mode (AV_MODE, MODE = 5)", page 72.
- Analog Velocity Gear mode, AVG_MODE. See "Analogue Velocity Gear Mode (AVG_MODE, MODE = 6)", page 72.
- TESTA_MODE. See "TESTA_MODE (MODE = 9)", page 74.
- HOME2_MODE. See "HOME2_MODE (MODE = 13)", page 76.
- HOME3_MODE. See "HOME3_MODE (MODE = 14)", page 77.
- Velocity Analogue Torque mode, VAT_MODE. See “Velocity Analogue Torque Mode, VAT_MODE (MODE = 17).” , page 79.
- Analogue Gear mode, AG_MODE. See “Analogue Gear Mode, AG_MODE (MODE = 18)” , page 79.
- Coil mode, COIL_MODE. See “Coil Mode, COIL_MODE (MODE = 19)” , page 79.
- Analogue 2 Position mode, A2POS_MODE. See “Analogue 2 Position mode, A2POS_MODE (MODE = 20)” , page 82.
- Analogue Position mode, APOS_MODE. See “Analogue Position mode, APOS_MODE (MODE = 21)” , page 82.
- Gear Follow mode, GF_MODE. See “Gear Follow Mode, GF_MODE (MODE = 24)” , page 83.

Full torque corresponds to a value of 1023.

3.2.8 P_FNC - Register 8

3.2.9

Data format: Longint.

Range: +/-2147483647

This register is updated by a function generator in the following modes:

- Velocity mode, V_MODE. See "Velocity function block", page 54 and "Velocity Mode", page 68.
- Position mode, P_MODE. See "Position function block", page 55 and "Position Mode", page 69.
- Analog velocity mode, AV_MODE. See "AV function block", page 58 and "Analog Velocity Mode (AV_MODE, MODE = 5)", page 72.
- Gear mode, G_MODE. See "Gear Follow Function block", page 57 and "Gear Position Mode (GP_MODE, MODE = 3)", page 69.
- Analog velocity gear mode, AVG_MODE. See "AVG function block", page 59 and "Analogue Velocity Gear Mode (AVG_MODE, MODE = 6)", page 72.
- Analogue Gear mode, AG_MODE. See "Analogue Gear Mode, AG_MODE (MODE = 18)", page 79. See “Analogue Gear Function block.” , page 60.
- Coil Mode, COIL_MODE. See “Coil Mode, COIL_MODE (MODE = 19)” , page 79.

Unit: 1/16 encoder counts.

P_FNC specifies the position that a given function generator has reached. The value of P_FNC is only used in P_MODE. The fact that the value is updated in other modes of operation enables the start position to be found when switched to P_MODE.

Example (flying saw):

1. The motor (saw) is stationary at its start position in P_MODE, while a target job passes.
2. At a given signal, the operating mode is switched to G_MODE. The motor (saw) synchronises velocity with the object. During this operation, P_FNC is updated.
3. The object is sawn.
4. The mode of operation is switched to P_MODE. Since the value of P_FNC has changed, the position function block will generate a velocity that moves the motor (saw) back to its starting point.

P_FNC can be used to perform relative positioning. See "Position function block", page 55. See “ERR_STAT - Register 35” , page 31., bit RELPOSPFNC.

3.2.10 P_IST - Register 10

Data format: LongInt.

Range: +/-2147483647.

This register indicates the motor’s current position.

Unit: Encoder counts

The register can be freely overwritten by a user. This will not influence control of the motor since the register contents are not used by the control program. This gives the user the facility to reset P_IST at a given position and thereafter measure the distance to another position, e.g. in conjunction with a linear guide.

Software position limits refer to this register. See "Software end-of-travel limits", page 86.

The register is updated in all modes of operation.

3.2.11

3.2.12 V_IST - Register 12

Data format: Integer.

Range: +/-32767.

This register indicates the motor velocity measured in encoder counts per sample. the sampling frequency is 520.8 Hz. The number of encoder counts per revolution = 4096.

If V_IST = 100, this corresponds to a motor velocity of:

$$\frac{100 * 520.8}{4096} = 12.715 \text{ rev/sec} = 762.9 \text{ rpm.}$$

The register should not be overwritten by the user since this will result in a positioning error.

The register is updated in all modes of operation.

Note: The resolution of the measured velocity stored in this register is approximately 8 rpm. Reading this register will therefore involve measurement noise (quantisation noise) amounting to +/- 8 rpm. In the context of control, this error is of no consequence since it is only valid for 1/520.8 sec = 1.92 ms.

Unit: 7.629 rpm.

3.2.13 KVOUT - Register 13

Data format: Fixed8

Recommended range: 1.0 ... 10.0

This register is used to determine a relative, inertial motor load, seen in relation to the motor's nominal inertial load. The calculation of KVOUT incorporates the motor's own inertia.

Example: Let a motor's own inertia be 17.3E-6 kgm2, (nominal load). The motor is loaded with an inertial load = 50E-6 kgm2.

KVOUT is calculated as:

$$\frac{17.3E\text{-}6 + 50E\text{-}6}{17.3E\text{-}6} = 3.9$$

During control, the calculated nominal torque will be multiplied by KVOUT, thus maintaining the motor acceleration.

KVOUT is used in the following modes of operation:

- Velocity mode. See "Velocity Mode", page 68.
- Position mode. See "Position Mode", page 69.
- Gear Position mode. See "Gear Position Mode (GP_MODE, MODE = 3)", page 69.
- Analog Velocity mode. See "Analog Velocity Mode (AV_MODE, MODE = 5)", page 72.
- Analog Velocity Gear mode. See "Analogue Velocity Gear Mode (AVG_MODE, MODE = 6)", page 72
- Stop mode. See "STOP_MODE (MODE = 11)", page 75.
- HOME1 mode. See "HOME1_MODE (MODE = 12)", page 75.
- HOME2 mode. See "HOME2_MODE (MODE = 13)", page 76.
- HOME3 mode. See "HOME3_MODE (MODE = 14)", page 77.
- Analogue Velocity mode. See "Analog Velocity Mode (AV_MODE, MODE = 5)", page 72
- Velocity Analogue Torque mode. See "Velocity Analogue Torque Mode, VAT_MODE (MODE = 17).", page 79.
- Analogue Gear mode. See "Analogue Gear Mode, AG_MODE (MODE = 18)", page 79.
- Coil mode. See "Coil Mode, COIL_MODE (MODE = 19)", page 79.
- Analogue 2 Position mode. See "Analogue 2 Position mode, A2POS_MODE (MODE = 20)" , page 82.
- Analogue Position mode. See "Analogue Position mode, APOS_MODE (MODE = 21)", page 82.
- Gear Follow mode. See "Gear Follow Mode, GF_MODE (MODE = 24)" , page 83.

See "Speed Regulator", page 64.

3.2.14 GEARF1 - Register 14

3.2.15 GEARF2 - Register 15

Data format, GEARF1: Integer.
Data format, GEARF2: Word.

These registers are used for electronic gearing of the signal at the pulse input.

Conversion ratio:

$$\frac{\text{Output}}{\text{Input}} = \frac{\text{GEARF1}}{\text{GEARF2}}$$

The direction of rotation of the motor can be reversed by changing the sign of GEARF1.

GEARF1 and GEARF2 are used in the following modes of operation:

- Gear Position mode, GP_MODE. See "Gear Position Mode (GP_MODE, MODE = 3)", page 69.
- Analog Velocity Gear mode, AVG_MODE. See "Analogue Velocity Gear Mode (AVG_MODE, MODE = 6)", page 72.
- Analogue Gear mode. See "Analogue Gear Mode, AG_MODE (MODE = 18)", page 79.
- Gear Follow mode, GF_MODE. See “Gear Follow Mode, GF_MODE (MODE = 24)” , page 83.

It is recommended that the gearing ratio, (fraction), is reduced:

$$\frac{\text{Output}}{\text{Input}} = \frac{15500}{10000} = \frac{31}{20}$$

3.2.16 I2T - Register 16
3.2.17 I2TLIM - Register 17

Data format: Word.

These registers are used for monitoring thermal overload of the motor as a result of current heat dissipation in the coils.

In principle, I2T carries out a partial integration of the current heat loss:

$Ia^2 + Ib^2 + Ic^2$

where Ia, Ib and Ic are the measured phase currents.

When the value of I2T exceeds I2TLIM the following occurs:

- I2T_ERR bit is set. See "ERR_STAT - Register 35", page 31.
- MODE_REG = INIT_MODE. See "STOP_MODE (MODE = 11)", page 75 and "Init Mode", page 67.

Register I2T is updated in all modes of operation.

3.2.18 UIT - Register 18
3.2.19 UITLIM - Register 19

Data format: Word.

These registers are used for monitoring thermal overload of the motor's internal power dump as a consequence of regenerative energy from the motor during deceleration.

In principle a partial integration of the regenerative power is carried out:

Preg = - (Ia*Ua + Ib*Ub + Ic*Uc)

where Ia, Ib and Ic are the measured phase currents, and Ua, Ub and Uc are the applied voltages to the three phases.

If Preg is calculated to be < 0, Preg is set = 0.

When the value of UIT exceeds UITLIM, and the supply voltage > 52V, the following occurs:

- UIT_ERR bit is set. See "ERR_STAT - Register 35", page 31.
- MODE_REG = INIT_MODE. See "STOP_MODE (MODE = 11)", page 75 and "Init Mode", page 67.

Register UIT is updated in all modes of operation.

3.2.20 FLWERR - Register 20

3.2.21

3.2.22 FLWERRMAX - Register 22

(Follow Error)
Data format: LongInt.

Range, FLWERR: +/-67E6.
Range, FLWERRMAX: 0 ... 67E6.

These registers are used in the following modes of operation:

- Velocity mode, V_MODE. See "Velocity Mode", page 68.
- Position mode, P_MODE. See "Position Mode", page 69.
- Gear Position mode, GP_MODE. See "Gear Position Mode (GP_MODE, MODE = 3)", page 69.
- Analog Velocity mode, AV_MODE. See "Analog Velocity Mode (AV_MODE, MODE = 5)", page 72.
- Analog Velocity Gear mode, AVG_MODE. See "Analogue Velocity Gear Mode (AVG_MODE, MODE = 6)", page 72.
- Velocity Analogue Torque mode, VAT_MODE. See "Velocity Analogue Torque Mode, VAT_MODE (MODE = 17)."", page 79.
- Analogue Gear mode. See "Analogue Gear Mode, AG_MODE (MODE = 18)", page 79
- Coil mode. See "Coil Mode, COIL_MODE (MODE = 19)", page 79
- Analogue 2 Position mode. See “Analogue 2 Position mode, A2POS_MODE (MODE = 20)” , page 82.
- Analogue Position mode. See "Analogue Position mode, APOS_MODE (MODE = 21)", page 82.
- Gear Follow mode, GF_MODE. See “Gear Follow Mode, GF_MODE (MODE = 24)” , page 83.

In the above modes, FLWERR is calculated. In all other modes of operation FLWERR is set to 0.

V, AV, AVG, AVZ, VAT_MODE: In these modes of operation FLWERR is calculated by accumulating velocity errors that are in part due to the dynamic characteristics of the velocity regulator, and in part can be due to the fact that the motor is torque overloaded, which results in the expected position cannot be maintained. FLWERR is thus used here to detect abnormal loading of the motor or blockages.

P, GP, AG,_COIL, A2POS, APOS, GF_MODE: FLWERR is calculated in the same way as described above. It should be noted however that in these modes the function generator will correct/adjust the velocity profile so that the motor is not significantly overloaded. The value of FLWERR will therefore be limited automatically. The user can however still use FLWERR, FLWERRMAX in applications in which the maximum motor torque is expected to be so small that a correction of the velocity profile will not take place, and where the user wants to use FLWERR as an indicator of an abnormal torque sequence.

If the absolute value of FLWERR exceeds FLWERRMAX, the following occurs:

- FLW_ERR bit is set. See "ERR_STAT - Register 35", page 31. "Init Mode", page 67.

This monitoring function can be disabled by setting FLWERRMAX = 0.

See "Follow error", page 87.

3.2.23

3.2.24 FNCERR

3.2.25

3.2.26 FNCERRMAX

(Function Error).

Data format: LongInt.

Range, FNCERR: +/-134217727.
Range, FNCERRMAX: 0 ... 134217727.

These registers are used in the following modes of operation:

- Velocity mode, V_MODE. See "Velocity Mode", page 68.
- Position mode, P_MODE. See "Position Mode", page 69.
- Gear mode, G_MODE. See "Gear Position Mode (GP_MODE, MODE = 3)", page 69.
- Analogue Velocity mode. See "Analog Velocity Mode (AV_MODE, MODE = 5)", page 72.
- Analogue Velocity Gear mode. See "Analogue Velocity Gear Mode (AVG_MODE, MODE = 6)", page 72.
- Velocity Analogue Torque mode. See "Velocity Analogue Torque Mode, VAT_MODE (MODE = 17).", page 79.
- Analogue Gear mode. See "Analogue Gear Mode, AG_MODE (MODE = 18)", page 79
- Coil mode. See "Coil Mode, COIL_MODE (MODE = 19)", page 79.
- Analogue 2 Position mode. See “Analogue 2 Position mode, A2POS_MODE (MODE = 20)” , page 82.
- Analogue Position mode. "Analogue Position mode, APOS_MODE (MODE = 21)", page 82.

In the above modes FNCERR is calculated. In all other modes of operation FNCERR is set to 0.

FNCERR is calculated by accumulating the velocity corrections that the function generator has applied to the velocity profile in order to maintain the specified maximum torque.

It should be noted that the value of FNCERR in P_MODE, and related modes, is not an expression of a positioning error. If the motor is forced from its correct position and stays there, the function generator will generate a velocity towards the correct position. Assume for example that this velocity = 10. Since the motor is kept in this position, in principle the velocity profile will be corrected by a value of -10 at each sample. The absolute value of FNCERR will thus increase by 10 for each sample, despite the fact that the positioning error is constant.

GP, GF_MODE: FNCERR is calculated by accumulating the velocity corrections that the function generator has applied to the velocity profile in order to maintain the specified maximum torque, maximum acceleration and maximum velocity. In GP_MODE, and related modes, FNCERR represents a positioning error.

If the absolute value of FNCERR exceeds FNCERRMAX, the following occurs:

- FNC_ERR bit is set. See "ERR_STAT - Register 35", page 31.
- MODE_REG = INIT_MODE. See "STOP_MODE (MODE = 11)", page 75 and "Init Mode", page 67.

This monitoring function can be disabled by setting FNCERRMAX = 0.

In P_MODE especially, it is normally unacceptable that FNCERR deviates from 0, since this indicates that the operating parameters have been set in such a way that the motor is physically unable to follow the required velocity profile. The specified maximum torque, T_SOLL, and maximum acceleration, A_SOLL, should be corrected by the user.

See "Function Error", page 87.

3.2.27

3.2.28 MIN_P_IST - Register 28

3.2.29

3.2.30 MAX_P_IST - Register 30

Data format: LongInt.

Range: +/-2147483647.

These registers are used as software limit stops.
If:

- P_IST > MAX_P_IST or
- P_IST < MIN_P_IST

the following occurs:

- PLIM_ERR bit is set. See "ERR_STAT - Register 35", page 31.
- MODE_REG = INIT_MODE. See "STOP_MODE (MODE = 11)", page 75 and "Init Mode", page 67.

The monitoring function can be disabled by setting:

- MAX_P_IST = 0
- MIN_P_IST = 0

See "Software end-of-travel limits", page 86.

3.2.31 Reserved

3.2.32 ACC_EMERG - Register 32

Data format: Word.

Recommended range: 1 ... 1000.

This register is used in the event that the motor develops an error condition. See "Monitoring Functions", page 85, "Stop function block", page 59, and "STOP_MODE (MODE = 11)", page 75.

The specified maximum acceleration, A_SOLL, is replaced by the value of ACC_EMERG during deceleration. See "A_SOLL - Register 6", page 20.
Use of ACC_EMERG can be disabled by setting ACC_EMERG = 0.

3.2.33 INPOSWIN - Register 33

3.2.34 INPOSCNT - Register 34

Data format: Word.

Range: 0 ... 32767.

In Position mode, Gear mode and related modes:

Via these registers, the user can specify conditions to define when the motor is in position:

- INPOSWIN specifies the number of encoder counts from which the motor position may deviate from the correct position. Note that the motor is only in position when the deviation in position is *less than* the value of INPOSWIN. If the deviation in position is *equal to* the value of INPOSWIN, the motor is *not* in position.
- INPOSCNT specifies the number of successive samples for which this tolerance must be maintained.

The purpose of INPOSCNT is to ensure that an overshoot in position, in which a sampled position is within the interval specified by INPOSWIN, does not result in an indication that the correct position has been reached.

Once the motor is in position according to the specified conditions, the following occurs:

- IN_POS bit is set in accordance with conditions that are dependent on the mode of operation. See "Position Mode", page 69, "Gear Position Mode (GP_MODE, MODE = 3)", page 69, and "ERR_STAT - Register 35", page 31.

Recommended value for INPOSCNT: 3 – 5.

In Velocity mode and related modes:

Via these registers, the user can specify conditions to define when the motor velocity corresponds to the required velocity, or when the motor is stationary:

- INPOSWIN specifies the number of encoder counts per sample that the motor velocity may deviate by from either the required velocity or from the stationary state. See "V_SOLL - Register 5", page 19 and "V_IST - Register 12", page 23.

- If INPOSCNT > 0, INPOSCNT specifies the number of consecutive samples that the motor must correspond to the desired velocity with the tolerance specified in INPOSWIN before the IN_POS bit is set. If INPOSCNT = 0, INPOSWIN specifies the tolerance for the stationary condition, within which the IN_POS bit is set. See "ERR_STAT - Register 35", page 31.

Recommended value for INPOSCNT: 0, 3 – 5.

3.2.35 ERR_STAT - Register 35

Data format: Word.
This register contains a number of status and error bits:

- Bit 14: DIS_P_LIM. If this bit is set, monitoring of the software position's end-of-travel limit will be disabled. This feature can be used for example in conjunction with homing modes, in which the homing position often lies outside the normally allowed operating range. See "MIN_P_IST - Register 28" , page 29. See "HOME1_MODE (MODE = 12)" , page 75. See "HOME2_MODE (MODE = 13)" , page 76. See "HOME3_MODE (MODE = 14)" , page 77.
- Bit 13: UV_DETECT: If this bit is set, the supply voltage has been measured to be lower than U_MIN_SUP. See "U_MIN_SUP - Register 152" , page 49.
- Bit 12: UV_ERR. If this bit is set, an undervoltage error has been detected. See "UV_HANDLE - Register 160" , page 50. See "Undervoltage detection" , page 87.
- Bit 11: IX_ERR. In order to regulate the motor's three phase currents, the phase currents are measured and converted to digital values that are used as the basis for calculating phase voltages. The measurement range for phase currents is limited by a minimum and maximum value. If a current value exceeds the measurement range, the value will be represented by the corresponding minimum or maximum value, which in terms of current regulation constitutes a critical condition. Therefore this bit is set if a phase current is measured as the minimum or maximum value.
- Bit 10: RELPOSPFNC. If this bit is set, positioning using the positioning registers will be regarded as a relative positioning, which is carried out as follows: P_FNC = P_FNC - 16 * Px. See "Position function block", page 55, "P_FNC - Register 8", page 22, "P_REG_P - Register 43", page 36 and "Format", page 93, command-mode. If this bit is reset, see bit 9: RELPOSPSOLL.
- Bit 9: RELPOSPSOLL: If this bit is set, positioning using the positioning registers will be regarded as a relative positioning: P_SOLL = P_SOLL + Px. See "Position function block", page 55, "P_SOLL - Register 3", page 18, "P_REG_P - Register 43", page 36 and "Format", page 93, command mode. If this bit is reset and bit 0: RELPOSPFNC is reset, positioning will be carried out absolutely via: P_SOLL = Px.
- Bit 8: FRAME_ERR_TX. See "Synchronisation", page 97.
- Bit 7: PLIM_ERR: Software position limit exceeded. Mode is automatically switched to Init-mode when this error occurs See "Software end-of-travel limits", page 86.
- Bit 6: DEC_FLAG: The motor is under deceleration. See "Velocity function block", page 54 and "Position function block", page 55
- Bit 5: ACC_FLAG: The motor is under acceleration. See "Velocity function block", page 54 and "Position function block", page 55.
- Bit 4: IN_POS: The motor is"in position". See "Velocity Mode", page 68, "Position Mode", page 69, "Gear Position Mode (GP_MODE, MODE = 3)", page 69, "INPOSWIN - Register 33", page 30, and "INPOSCNT - Register 34", page 30.

- Bit 3: UIT_ERR: The internal power-dump circuitry is overloaded. Use external circuitry. Mode is automatically switched to Init-mode when this error occurs. See "UIT - Register 18", page 26 and "UIT", page 86.
- Bit 2: FNC_ERR: FNCERR exceeds FNCERRMAX. Mode is automatically switched to Init-mode when this error occurs. See "FNCERR", page 28 and "Function Error", page 87.
- Bit 1: FLW_ERR: FLWERR exceeds FLWERRMAX. Mode is automatically switched to Init-mode when this error occurs. See "FLWERR - Register 20", page 27 and "Follow error", page 87.
- Bit 0: I2T_ERR: The motor is thermally overloaded (calculated coil temperature). Mode is automatically switched to Init-mode when this error occurs. See "I2T - Register 16", page 26 and "I²T", page 86.

If an error bit is set (bit 0 .. 3) and a mode switch is made to Init-mode, this error bit must be reset before a switch from Init-mode can be made. See "Init Mode", page 67.

3.2.36 CNTRL_BITS - Register 36

Data format: word.

This register contains a number of control and status bits:

- Bit 15 .. 13: Status for hall-signals. See "ELDEGP_OFFSET - Register 125", page 45.
- Bit 12: REL_RESYNC: Relative re-synchronisation of position. See bit 10: MAN_RESYNC.
- Bit 11: INDEX_HOME: The motor's encoder does not include an index marker. Instead, the transition between two Hall conditions is intended as an index marker. Since the motor contains two pole-pairs, two index markers are thus defined per revolution. The index markers are used in conjunction with HOMING-modes to provide more accurate determination of the home-position. If INDEX_HOME = 1 in a homing mode, each read value of the home position will be rounded to the nearest index marker.
- Bit 10: MAN_RESYNC. When this bit is set, a manual re-synchronisation of position is carried out. The bit is automatically reset once the re-synchronisation has been completed. The re-synchronisation is relative if bit 12: REL_RESYNC is set. The re-synchronisation is absolute if bit 12: REL_RESYNC is reset.
- Bit 9: AUTO_RESYNC. When this bit is set, an automatic re-synchronisation of position is carried out every time the motor is in INIT_MODE.This re-synchronisation is always absolute.
- Bit 8: RECINNERBIT (Record Inner loop)
  For data sampling to the sample buffer, either the sampling frequency for current loops (inner loop) = 7812 Hz or the sampling frequency for velocity-/position loops = 520.8 Hz can be used. If RECINNERBIT is set (1), a sampling frequency of 7812 Hz for data acquisition. See "SAMPLE1 - Register 112", page 43, "Operation", page 13, "TESTU_MODE (MODE = 8)", page 73, and "TESTA_MODE (MODE = 9)", page 74.

- Bit 7: REWINDBIT:
  If this bit is set (1), the sample buffer pointer is moved to the start of the sample buffer. (REC_CNT = 0). The bit is automatically reset once the operation has been performed. See "Operation", page 13, and "REC_CNT. - Register 116", page 43.
- Bit 6: RECORDBIT:
  If this bit is set (1), data sampling to the sample buffer is started. The bit is reset automatically when the sample buffer is full. See "SAMPLE1 - Register 112", page 43, "REC_CNT. - Register 116", page 43, "Operation", page 13, "TESTU_MODE (MODE = 8)", page 73, and "TESTA_MODE (MODE = 9)", page 74.
- Bit 5: HALL_INT: For correction of the instantaneous electrical angle, ELDEG_IST, an interrupt request is given when hall elements change between two specific states. During normal operating conditions, this correction is only necessary once immediately after power on: The first correction occurs automatically, after which the hall-interrupt is disabled. If HALL_INT is set (1), the hall-interrupt remains enabled. This is used in conjunction with setting ELDEGN_OFFSET and ELDEGP_OFFSET. See "ELDEGN_OFFSET - Register 124", page 45 and "ELDEGP_OFFSET - Register 125", page 45.
- Bit 4: HICLK:
  The Pulse Inputs are connected to a digital filter in order to eliminate noise. If HICLK is set (1), the maximum frequency that may pass through the filter is 1 MHz. If HICLK is reset, the maximum frequency is 31 KHz. This is valid providing that the duty cycle of the input signal is 50%. This bit is only used in conjunction with reset/power up of the motor. A change of the bit status first takes effect after write to flash / reset. See "Operation", page 13.
- Bit 3: INPSIGN:
  This bit changes the sign of the input signal in terms of hardware. This bit is only used in conjunction with reset / power up of the motor. A change of the status of this bit therefore only has effect after write to flash / reset. See "Operation", page 13.
- Bit 2: PULSEDIR:
  This bit selects the signal format of the input signal. If the bit is set (1), the format: Pulse + direction is selected. If the bit is reset, quadrature signal is selected. This bit is only used in conjunction with reset / power up of the motor. A change of the status of this bit therefore only has effect after write to flash / reset. See "Operation", page 13.
- Bit 1,0: USERINTF1, USERINTF0:
  These bits are used to set the function of the user interface:

  00: Signal input. (Pulse + direction / quadrature signal).
  01: Output from internal encoder. (Quadrature signal.)
  11: Communication. (FASTMAC / FLEXMAC protocol).

  These bits are only used in conjunction with reset / power up of the motor. A change of the status of these bits therefore only has effect after write to flash / reset. See "Operation", page 13.

3.2.37 STARTMODE - Register 37

Data format: Word.

This register is used to indicate the mode in which the motor will start after power on and HOME_MODE. See "HOME_MODE - Register 42", page 35, "HOME1_MODE (MODE = 12)", page 75, "HOME2_MODE (MODE = 13)", page 76 and "HOME3_MODE (MODE = 14)", page 77.

If the HOME_MODE register is specified as 0, the motor is started directly in the mode specified by STARTMODE.

3.2.38 P_HOME - Register 38

3.2.39

Data format: LongInt.

Range: +/-67E6.

This register specifies the value of the home position, found in one of the three home-modes. See "HOME1_MODE (MODE = 12)", page 75, "HOME2_MODE (MODE = 13)", page 76 and "HOME3_MODE (MODE = 14)", page 77.

3.2.40 V_HOME - Register 40

Data format: Integer.

Recommended range: See "V_SOLL - Register 5", page 19.

This register specifies the motor velocity used during return to the home position in one of the three home modes. See "HOME1_MODE (MODE = 12)", page 75, "HOME2_MODE (MODE = 13)", page 76 and "HOME3_MODE (MODE = 14)", page 77.

The sign of V_HOME specifies the direction of the home search.

3.2.41 T_HOME - Register 41

Data format: Integer.

Range: -800 ... 800

The significance of this register is dependent on the home mode selected:

- HOME1_MODE:
  In this mode, a home search is made for a mechanical limit (mechanical blockage). The Home mode is defined as the position that the motor has reached when the torque specified by T_HOME is exceeded 50 times. The maximum allowable torque during home search is 1.25 times the value of T_HOME. Since the maximum torque limit is 1023, the maximum value of T_HOME is thus approx. 800. See "HOME1_MODE (MODE = 12)", page 75.
- HOME2, HOME3_MODE:
  During home search, the maximum torque specified by T_SOLL is used. In these modes, an electrical limit is used, connected to the analogue input. The sign of T_HOME determines the polarity of the input signal: the input signal is active high if the sign of T_HOME is positive; otherwise active low. See "HOME2_MODE (MODE = 13)", page 76 and "HOME3_MODE (MODE = 14)", page 77.

3.2.42 HOME_MODE - Register 42

Data format: Word, (2 bytes).

This register specifies the home mode that is used after power on (low byte) and the home mode used in conjunction with a home command via the FastMac-protocol, (high byte). See "Format", page 93, command-mode.

If the value is set to 0, home mode is ignored and the motor is started directly in the mode specified by STARTMODE. See "STARTMODE - Register 37", page 34.
Legal home modes are:

- 12: HOME1_MODE:
  A search for the home position is made until a mechanic limit is reached. See "HOME1_MODE (MODE = 12)", page 75.
- 13: HOME2_MODE:
  A search for the home position is made until an electrical home signal is activated. See "HOME2_MODE (MODE = 13)", page 76.
- 14: HOME3_MODE:
  A search for the home position is made until an electrical home signal is activated/ deactivated. See "HOME3_MODE (MODE = 14)", page 77.

Once the home position has been found, the motor is started in the mode specified by STARTMODE. See "STARTMODE - Register 37", page 34.

3.2.43 P_REG_P - Register 43

Data format: Word.

Range: 0 ... 8

This register is used as a pointer to one of eight position registers when the motor is controlled in register mode.
See "P1 - Register 49", page 38.

If the value of P_REG_P is specified as 3, the value of P3 will be copied or added to P_SOLL. At the same time, P_REG_P will be overwritten with the value 0, to indicate that the action has been performed.
See "P_SOLL - Register 3", page 18.

See RELPOSPSOLL and RELPOSPFNC bits, "ERR_STAT - Register 35", page 31.

3.2.44 V_REG_P - Register 44

Data format: Word.

Range: 0 ... 8

This register is used as a pointer to one of eight velocity registers when the motor is controlled in register mode.
See "V1 - Register 65", page 38.

If the value of V_REG_P is specified as 3, the value of V_SOLL will be overwritten by the value of V3. At the same time, V_REG_P will be overwritten with the value 0, to indicate that the action has been performed. See "V_SOLL - Register 5", page 19.

3.2.45 A_REG_P - Register 45

Data format: Word.

Range: 0 ... 4

This register is used as a pointer to one of 4 acceleration registers when the motor is controlled in register mode.
See "A1 - Register 73", page 39.

If the value of A_REG_P is specified as 3, the value of A_SOLL will be overwritten by the value of A3. At the same time, A_REG_P will be overwritten with the value 0, to indicate that the action has been performed. See "A_SOLL - Register 6", page 20.

3.2.46 T_REG_P - Register 46

Data format: Word.

Range: 0 ... 4

This register is used as a pointer to one of 4 torque registers when the motor is controlled in register mode.
If the value of T_REG_P is specified as 3, the value of T_SOLL will be overwritten by the value of T3. At the same time, T_REG_P will be overwritten with the value 0, to indicate that the action has been performed. See "T_SOLL - Register 7", page 21.

3.2.47 L_REG_P - Register 47

Data format: Word.

Range: 0 ... 4

This register is used as a pointer to one of 4 load registers (inertias) when the motor is controlled in register mode.

If the value of L_REG_P is specified as 3, the value of KVOUT will be overwritten by the value of L3. At the same time, L_REG_P will be overwritten with the value 0, to indicate that the action has been performed. See "KVOUT - Register 13", page 24.

3.2.48 Z_REG_P - Register 48

Data format: Word.

Range: 0 ... 4

This register is used as a pointer to one of 4 position tolerance registers when the motor is controlled in register mode.
If the value of Z_REG_P is specified as 3, the value of INPOSWIN will be overwritten by the value of Z3. At the same time, Z_REG_P will be overwritten with the value 0, to indicate that the action has been performed. See "INPOSWIN - Register 33", page 30 and "INPOSCNT - Register 34", page 30.

3.2.49 P1 - Register 49

3.2.51 P2 - Register 51

3.2.53 P3 - Register 53

3.2.55 P4 - Register 55

3.2.57 P5 - Register 57

3.2.59 P6 - Register 59

3.2.61 P7 - Register 61

3.2.63 P8 - Register 63

Data format: LongInt.

Range: +/-67E6

These 8 registers are used in register mode and can contain 8 different values of position. A position is transferred to P_SOLL using P_REG_P. See "P_REG_P - Register 43", page 36 and "P_SOLL - Register 3", page 18.

3.2.65 V1 - Register 65
3.2.66 V2 - Register 66
3.2.67 V3 - Register 67
3.2.68 V4 - Register 68
3.2.69 V5 - Register 69
3.2.70 V6 - Register 70
3.2.71 V7 - Register 71
3.2.72 V8 - Register 72

Data format: Integer.

Range: See "V_SOLL - Register 5", page 19.

These 8 registers are used in register mode and can contain 8 different velocity values. A velocity is transferred to V_SOLL using V_REG_P. See "V_SOLL - Register 5", page 19 and "V_REG_P - Register 44", page 36.

3.2.73 A1 - Register 73
3.2.74 A2 - Register 74
3.2.75 A3 - Register 75
3.2.76 A4 - Register 76

Data format: Word.

Recommended range: 1 ... 1000.

These 4 registers are used in register mode and can contain 4 different acceleration values. An acceleration is transferred to A_SOLL using A_REG_P. See "A_SOLL - Register 6", page 20 and "A_REG_P - Register 45", page 36.

3.2.77 T1 - Register 77
3.2.78 T2 - Register 78
3.2.79 T3 - Register 79
3.2.80 T4 - Register 80

Data format: Word.

Range: 0 ... 1023

These 4 registers are used in register mode and can contain 4 different torque values. A torque is transferred to T_SOLL using T_REG_P. See "T_SOLL - Register 7", page 21 and "T_REG_P - Register 46", page 37.

3.2.81 L1 - Register 81
3.2.82 L2 - Register 82
3.2.83 L3 - Register 83
3.2.84 L4 - Register 84

Data format: Fixed8.

Recommended range: 1.0 ... 10.0

These 4 registers are used in register mode and can contain 4 different values of inertial load factor. A load factor is transferred to KVOUT using T_REG_P. See "KVOUT - Register 13", page 24 and "L_REG_P - Register 47", page 37.

3.2.85 Z1 - Register 85
3.2.86 Z2 - Register 86
3.2.87 Z3 - Register 87
3.2.88 Z4 - Register 88

Data format: Word.

Range: 0 ... 32767.

These 4 registers are used in register mode and can contain 4 different values of position tolerance. A tolerance is transferred to INPOSWIN using Z_REG_P. See "INPOSWIN - Register 33", page 30, "INPOSCNT - Register 34", page 30 and "Z_REG_P - Register 48", page 37.

3.2.89 KFF3 - Register 89
3.2.90 KFF2 - Register 90
3.2.91 KFF1 - Register 91
3.2.92 KFF0 - Register 92

Data format: Fixed8.

Range: +/-127.996

These 4 registers contain coefficients for filter FF.

Filter transfer function:

$$\frac{\text{Output}(z)}{\text{Input}(z)} = \frac{KFF3\, z^3 + KFF2\, z^2 + KFF1\, z + KFF0}{z^3}$$

See "Speed Regulator", page 64.

3.2.93 KVFX4 - Register 93
3.2.94 KVFX3 - Register 94
3.2.95 KVFX2 - Register 95
3.2.96 KVFX1 - Register 96
3.2.97 KVFY3 - Register 97
3.2.98 KVFY2 - Register 98
3.2.99 KVFY1 - Register 99
3.2.100 KVFY0 - Register 100

Data format: Fixed4.

Range: +/-7.9997

These 8 registers contain coefficients for filter VF.

Filter transfer function:

$$\frac{\text{Output}(z)}{\text{Input}(z)} = \frac{KVFX4\, z^4 + KVFX3\, z^3 + KVFX2\, z^2 + KVFX1\, z}{z^4 - KVFY3\, z^3 - KVFY2\, z^2 - KVFY1\, z - KVFY0}$$

Note the change of sign in the transfer function.

See "Speed Regulator", page 64.

3.2 Parameter and Data Registers

3.2.101 GEARB - Register 101
3.2.102 KVB3 - Register 102
3.2.103 KVB2 - Register 103
3.2.104 KVB1 - Register 104
3.2.105 KVB0 - Register 105

Data format: Fixed8.

Range: +/-127.996

These 5 registers contain coefficients for filter VB.

Filter transfer function:

$$\frac{\text{Output}(z)}{\text{Input}(z)} = \text{GEARB} * \frac{KVB3\, z^3 + KVB2\, z^2 + KVB1\, z + KVB0}{z^3}$$

See "Speed Regulator", page 64.

3.2.106 KIFX2 - Register 106
3.2.107 KIFX1 - Register 107
3.2.108 KIFY1 - Register 108
3.2.109 KIFY0 - Register 109

Data format: Fixed4.

Range: +/-7.9997

These 4 registers contain coefficients for filter IF.

Filter transfer function:

$$\frac{\text{Output}(z)}{\text{Input}(z)} = \frac{KIFX2\, z^2 + KIFX1\, z}{z^2 - KIFY1\, z - KIFY0}$$

Note the change of sign in the transfer function.

See "Current Regulator", page 65.

Warning: Incorrect setting of these coefficients can cause irreparable damage and destroy the motor and circuitry since control of the phase currents may be lost.

3.2.110 KIB1 - Register 110

3.2.111 KIB0 - Register 111

Data format: Fixed8.

Range: +/-127.996

These 2 registers contain coefficients for filter IB.

Filter transfer function:

$$\frac{\text{Output}(z)}{\text{Input}(z)} = \frac{KIB1\ z + KIB0}{z}$$

See "Current Regulator", page 65.

Warning: Incorrect setting of these coefficients can cause irreparable damage and destroy the motor and circuitry since control of the phase currents may be lost.

3.2.112 SAMPLE1 - Register 112
3.2.113 SAMPLE2 - Register 113
3.2.114 SAMPLE3 - Register 114
3.2.115 SAMPLE4 - Register 115

Data format: Word.

Range: 1 ... 164

These registers are used in conjunction with real-time sampling to the sample buffer.

The number designation of the registers to be sampled to the buffer are specified in SAMPLE1 ... SAMPLE4.

See "CNTRL_BITS - Register 36", page 32, "TESTU_MODE (MODE = 8)", page 73 and "TESTA_MODE (MODE = 9)", page 74.

3.2.116 REC_CNT. - Register 116

Data format: Word.

Range: 0 ... 895

This register specifies the number of samples that are written in conjunction with sampling to the sample buffer, or the number of samples that are read by reading the buffer. The content of REC_CNT is reset when the REWIND-bit is set in CNTRL_BITS. See "CNTRL_BITS - Register 36", page 32.

3.2.117 FNC_OUT - Register 117

Data format: Integer.

This register contains output (velocity) from the function generator. See "Function blocks (Profile Generation)", page 54 and "Speed Regulator", page 64.

3.2.118 FF_OUT - Register 118

Data format: Integer.

This register contains output (velocity) from filter FF. See "Speed Regulator", page 64 and "KFF3 - Register 89", page 40.

3.2.119 VB_OUT - Register 119

Data format: Integer.

This register contains output (velocity) from filter VB. See "Speed Regulator", page 64, "GEARB - Register 101", page 41, and "KVB3 - Register 102", page 41.

3.2.120 V_EXT - Register 120

Data format: Integer.

This register contains the number of counts at the pulse input during the last sampling period.

Sampling frequency: 520.8 Hz

The register is updated in the following modes of operation:

- INIT_MODE. See "Init Mode", page 67.
- G_MODE. See "Gear Position Mode (GP_MODE, MODE = 3)", page 69.
- AVG_MODE. See "Analogue Velocity Gear Mode (AVG_MODE, MODE = 6)", page 72.
- AG_MODE. See “Analogue Gear Mode, AG_MODE (MODE = 18)” , page 79.
- COIL_MODE. See “Coil Mode, COIL_MODE (MODE = 19)” , page 79.

3.2.121 VF_OUT - Register 121

Data format: Integer.

This register contains output (torque) from filter VF. See "Speed Regulator", page 64.

Range: +/- 1023

3.2.122 ANINP - Register 122

Data format: Integer.

This register contains the measured voltage at the analogue input + ANINP_OFFSET. See "ANINP_OFFSET - Register 123", page 44.

The register is used in the following modes of operation:

- Analog Torque mode, AT_MODE. See "Analog Torque Mode (AT_MODE, MODE = 4)", page 70.
- Analog Velocity mode, AV_MODE. See "Analog Velocity Mode (AV_MODE, MODE = 5)", page 72.
- Analog Velocity Gear mode, AVG_MODE. See "Analogue Velocity Gear Mode (AVG_MODE, MODE = 6)", page 72.
- Home2 mode. See "HOME2_MODE (MODE = 13)", page 76.
- Home3 mode. See "HOME3_MODE (MODE = 14)", page 77.

The register is updated in all modes of operation.

During conversion of analogue voltages, +/- 10V is converted to +/- 1023.

3.2.123 ANINP_OFFSET - Register 123

Data format: Integer.

This register is used for compensating an offset voltage at the analogue input.

See "ANINP - Register 122", page 44.

3.2.124 ELDEGN_OFFSET - Register 124
3.2.125 ELDEGP_OFFSET - Register 125

Data format: Word.

Range: 0 ... 2047

For correct operation of the motor, it is necessary to know the electrical rotor angle. To measure this, a magnet that successively activates three hall elements during rotation is mounted on the rotor axle.

The zero-point for the electrical rotor angle is defined by the point at which the status for the hall signals changes between two distinct states. Electrical rotor angles other than zero are calculated by measurement of the number of encoder counts that the rotor has moved from the zero point.

The motor contains two pole pairs. There are 4096 encoder-counts per revolution. This gives 2048 encoder-counts per electrical period.

In practice, deviations exist from motor to motor in terms of when the hall signal status changes in relation to the rotor's physical electrical zero-point. To compensate for these deviations, the registers ELDEGN_OFFSET and ELDEGP_OFFSET are used.

The value of ELDEGN_OFFSET is written to ELDEG_IST when the status of hall signals changes in a negative direction of rotation. The value of ELDEGP_OFFSET is written to ELDEG_IST when the status of the hall signals changes in a positive direction.

See "PHASE_COMP - Register 126", page 45, "Current Regulator", page 65, and "CNTRL_BITS - Register 36", page 32. See "ELDEG_IST - Register 143" , page 48.

3.2.126 PHASE_COMP - Register 126

Data format: Fixed8.

Recommended range: 3.0 ... 7.0

The phase currents of the motor are regulated by three regulation loops with a given amplitude and phase characteristic. When the motor rotates, the three phase currents are sinusoidal, with a frequency dependent on the rate of rotation. At high rates of revolution, the electrical phase shift will be significant due to the regulator's phase characteristics.

This phase-shift is compensated for by adding an electrical angle:

V_ELDEG * PHASE_COMP

to the argument for the sine functions.

See "ELDEGN_OFFSET - Register 124", page 45, "ELDEGP_OFFSET - Register 125", page 45, "V_ELDEG - Register 144", page 49 and "Current Regulator", page 65.

3.2.127 AMPLITUDE - Register 127

Data format: Fixed8.

Range: 0 ... 0.83

This register specifies the maximum peak current/phase for the motor.

The measurement range for phase current is +/- 1023. If the required current in a phase is specified as this maximum value, any overshoot during regulation of this current will not be measured and the physical current in the phase cannot be controlled.

The regulation program calculates the maximum required phase current as follows:

1023 * AMPLITUDE

The maximum value of AMPLITUDE is thus restricted to 0.83, so that any overshoot in 1023 * 0.83 = 850 is measured.

Warning: Setting the AMPLITUDE to a value greater than 0.83 can cause irreparable damage and destroy the motor and circuitry.

See "Current Regulator", page 65 and "IA_SOLL - Register 133", page 47.

3.2.128 MAN_I_NOM - Register 128

3.2.129 MAN_ALPHA - Register 129

Data format: Word.

Range, MAN_I_NOM: 0 ... 1023
Range, MAN_ALPHA: 0 ... 2047

These registers are used in MANI_MODE for manual set-up of the motor's stator field.

In MANI_MODE the three phase currents will be determined by:

- IA_SOLL = MAN_I_NOM * sin(MAN_ALPHA) * AMPLITUDE
- IB_SOLL = MAN_I_NOM * sin(MAN_ALPHA + 120 elect. degrees) * AMPLITUDE
- IC_SOLL = MAN_I_NOM * sin(MAN_ALPHA + 240 elect. degrees) * AMPLITUDE

Units for MAN_I_NOM: Dependent on motor type.
Units for MAN_ALPHA: 360 / 2048 ELECT. DEGREES = 0.176 elect. degrees.

See "Current Regulator", page 65, "IA_SOLL - Register 133", page 47, and "AMPLITUDE - Register 127", page 46.

3.2.130 UMEAS - Register 130

Data format: Word.

Range: 0 ... 100

In conjunction with identification of the motor-coil transfer function, a voltage step function is applied to the motor coils.

UMEAS specifies the amplitude of this voltage step.

The register is used in TESTU_MODE. See "TESTU_MODE (MODE = 8)", page 73.

3.2.131 I_NOM - Register 131

3.2.132 PHI_SOLL - Register 132

Data format: Word.

Intended for debug purposes only.

3.2.133 IA_SOLL - Register 133

3.2.134 IB_SOLL - Register 134

3.2.135 IC_SOLL - Register 135

Data format: Integer.

Range: -850 ... 850

These data registers specify input values for the three current regulation loops.

See "Current Regulator", page 65 and "AMPLITUDE - Register 127", page 46.

3.2.136 IX_SELECT - Register 136

Data format: Word.

Intended for debug purposes only.

3.2.137 IA_IST - Register 137
3.2.138 IB_IST - Register 138
3.2.139 IC_IST - Register 139

Data format: Integer.

Range: -1023 ... 1023

These data registers specify measured values for the three phase currents. See "Current Regulator", page 65.

3.2.140 IA_OFFSET - Register 140
3.2.141 IB_OFFSET - Register 141
3.2.142 IC_OFFSET - Register 142

Data format: Word.

Range: 0 ... 2047. Typical: 1000 ... 1050

These data registers contain offset values in conjunction with measurement of the three phase currents.

In INIT_MODE the three phase voltages are set to 0 Volt, and it is therefore assumed that the phase currents are zero in this mode. See "Init Mode", page 67.

In INIT_MODE, IA, IB and IC_OFFSET are adjusted so that:

- IA_IST = Ia_measured + IA_OFFSET = 0.
- IB_IST = Ib_measured + IB_OFFSET = 0.
- IC_IST = Ic_measured + IC_OFFSET = 0.

3.2.143 ELDEG_IST - Register 143

Data format: Word.

Range: 0 ... 2047

This data register specifies the rotor's instantaneous electrical angle.

Unit: 360 / 2048 electr. degrees = 0.176 electr. degrees.

See "Current Regulator", page 65, "ELDEGN_OFFSET - Register 124", page 45, "ELDEGP_OFFSET - Register 125", page 45, and "CNTRL_BITS - Register 36", page 32.

3.2.144 V_ELDEG - Register 144

Data format: Integer.

This data register indicates the motor's rate of revolution, measured at a sampling frequency: 7812.5 Hz.

Unit: Encoder counts / sampling period.

See "PHASE_COMP - Register 126", page 45 and "Current Regulator", page 65.

3.2.145 UA_VAL - Register 145
3.2.146 UB_VAL - Register 146
3.2.147 UC_VAL - Register 147

Data format: Integer.

Intended for debug purposes only.

3.2.148 KIA - Register 148
3.2.149 KIB - Register 149
3.2.150 KIC - Register 150

Data format: Word.

These registers are used for compensation of the amplification in the measurement circuits for the three phase currents. Nominal value = 256.

3.2.151 U_SUPPLY - Register 151

Data format: Word.

This data register indicates the measured supply voltage.

Unit: 53.7 mV

See "Current Regulator", page 65.

3.2.152 U_MIN_SUP - Register 152

Data format: Word.

This data register specifies the minimum supply voltage. If the supply voltage is measured to be below the specified minimum, the UV_DETECT bit is set. See "ERR_STAT - Register 35" , page 31. See "U_SUPPLY - Register 151" , page 49. See "Undervoltage detection" , page 87. See "UV_HANDLE - Register 160" , page 50.

3.2.153 MOTOR_TYPE - Register 153

Data format: Word.

This register indicates the motor type.

3.2.154 SERIAL_NMB - Register 154

Data format: LongInt.

This data register indicates the motor's serial number.

3.2.156 MYADDR - Register 156

Data format: Word.

Range: 1 ... 254

This register indicates the motor's address, when it is included in a multi-drop net with a master and a number of slaves.

See "MacTalk Protocol", page 90.

3.2.157 HW_VERSION - Register 157

Data format: word.

This register indicates changes in hardware due to new hardware versions. This information is used by software only.

3.2.158 CHKSUM - Register 158

Data format: longword.

The content of this register is used to ensure program and data validity.

3.2.160 UV_HANDLE - Register 160

Data format: word.

This register is used for setting up the function for the detection of undervoltage conditions. See "U_SUPPLY - Register 151" , page 49. See "U_MIN_SUP - Register 152" , page 49. See "Undervoltage detection" , page 87. The register contains the following bits:

Bit 0: SET_UV_ERR: If this bit is set, the UV_ERR bit is set in the event of an undervoltage detection. See "ERR_STAT - Register 35" , page 31.

Bit 1: UV_GO_PASSIVE: If this bit is set, a switch to INIT_MODE is made in the event of an undervoltage detection. See "MODE_REG - Register 2" , page 17. See "Init Mode" , page 67.

Bit 2: Reserved.

Bit 3: UV_SOLL0: If this bit is set, the maximum velocity is set, V_SOLL = 0, (motor is stopped). See “V_SOLL - Register 5” , page 19.

3.2.161 INV_OUTPUT - Register 161

Ask JVL for further details.

3.2.162 INDEX OFFSET - Register 162

Ask JVL for further details.

3.2.163 P_NEW - Register 163

3.2.164

This register contains a positional value that is used during re-synchronisation of position. See “CNTRL_BITS - Register 36” , page 32. Register 36 contains the following bits for re-synchronisation of position: MAN_RESYNC, AUTO_RESYNC and REL_RESYNC. Re-synchronisation can be described as follows:

```
IF  ( MAN_RESYNC AND NOT REL_RESYNC )  THEN
      P_SOLL = P_NEW
      P_IST = P_NEW
      P_FNC = P_NEW * 16
IF  ( MAN_RESYNC AND REL_RESYNC ) )  THEN
      P_SOLL = P_SOLL + P_NEW
      P_IST = P_IST + P_NEW
      P_FNC = P_FNC + P_NEW * 16
IF  ( ( MODE = INIT_MODE ) AND  AUTO_RESYNC )  THEN
      P_FNC = 16*P_IST
```

See “P_SOLL - Register 3” , page 18. See “P_IST - Register 10” , page 22. See “P_FNC - Register 8” , page 22. See “Position Mode” , page 69.

4 Program / Function Description

The program software contains:

- Function blocks for generation of velocity profiles.
  See "Function blocks (Profile Generation)", page 54.
- A velocity regulator. See "Speed Regulator", page 64.
- Current Regulator for the three phase currents. See "Current Regulator", page 65.
- Monitoring functions. See "Monitoring Functions", page 85.

"Function blocks (Profile Generation)", page 54 describes the various function blocks.

"Speed Regulator", page 64 explains the velocity/position regulator.

"Current Regulator", page 65 describes the current regulator.

The software controls the MAC-motor in several different modes of operation. "Modes", page 66 describes how the function blocks, regulation loops and parameters are used in the various modes.

The software also includes certain monitoring functions. These are described in "Monitoring Functions", page 85.

4.1 Function blocks (Profile Generation)

The software includes 7 different function blocks for generation of velocity profiles or torque profiles:

- Velocity function block. See "Velocity function block", page 54.
- Position function block. See "Position function block", page 55.
- Gear Follow Function block. See “Gear Follow Function block” , page 57.
- Torque function block. See "Torque function block", page 58.
- AV-function block. See "AV function block", page 58.
- AVG-function block. See "AVG function block", page 59.
- Stop-function block. See "Stop function block", page 59.
- AG-function block. See “Analogue Gear Function block.” , page 60.
- Analogue 2 Position function block. See “Analogue 2 Position function block” , page 60.
- Analogue Position Function block. See “Analogue Position function block” , page 61.
- Gear Position Function block. See “Gear Position Function block” , page 62.

The following sections explain the function of each of these blocks.

4.1.1 Velocity function block

This function block generates a velocity profile with the following parameters:

- V_SOLL: Required velocity.
- A_SOLL: Maximum allowable acceleration / deceleration to achieve the required velocity.

The following registers are updated:

- Velocity profile is written to register: FNC_OUT.
  See "FNC_OUT - Register 117", page 43.
- ACC_FLAG, DEC_FLAG and IN_POS flag. See "ERR_STAT - Register 35", page 31.
- FNC_ERR. See "FNCERR", page 28.

The function can be described by:

- DV = V_SOLL – V_OLD
- Limit DV to +/- A_SOLL
- FNC_OUT = V_OLD + DV
- ACC_FLAG = FNC_OUT > V_OLD.
- DEC_FLAG = FNC_OUT < V_OLD
- V_OLD = FNC_OUT
- If INPOSCNT=0 then IN_POS = | V_IST | < INPOSWIN, (motor stopped).
- If INPOSCNT>0 then IN_POS = | V_SOLL - V_IST*16 | < INPOSWIN, (velocity error < INPOSWIN). See “INPOSWIN - Register 33” , page 30. See “INPOSCNT - Register 34” , page 30.

Registers DV and V_OLD are not accessible to the user. V_OLD is set to 0 in INIT_MODE.

4.1 Function blocks (Profile Generation)

4.1.2 Position function block

This function block generates a velocity profile with the following parameters:

- P_SOLL: Required absolute position.
- V_SOLL: Maximum velocity to achieve the required position.
- A_SOLL: Maximum allowable acceleration / deceleration to achieve the required position.

The following registers are updated:

- Velocity profile is written to register: FNC_OUT.
  See "FNC_OUT - Register 117", page 43.
- ACC_FLAG and DEC_FLAG. See "ERR_STAT - Register 35", page 31.
- P_FNC. See below. See "P_FNC - Register 8", page 22.

The function can, in principle, be described by:

- T = V_OLD / A_SOLL, (calculated deceleration time).
- DP is calculated as the distance to the required position. (P_SOLL – P_FNC / 16).
- FNC_OUT = 2 * DP / T.
- Limit FNC_OUT to +/- V_SOLL.
- DV = V_SOLL – V_OLD
- Limit DV to +/- A_SOLL.
- FNC_OUT = V_OLD + DV
- ACC_FLAG = FNC_OUT > V_OLD.
- DEC_FLAG = FNC_OUT < V_OLD
- V_OLD = FNC_OUT
- P_FNC = P_FNC + FNC_OUT

Registers T, DP, DV and V_OLD are not accessible to the user. V_OLD is set to 0 in INIT_MODE.

The function is assigned a register, P_FNC, (Function position). This register is used to accumulate values in the calculated velocity profile, so that P_FNC contains the absolute position that the *function block* has reached. See "P_FNC - Register 8", page 22.

If the motor is overloaded mechanically, such that the torque necessary to maintain a given velocity profile exceeds T_SOLL, the velocity profile is ”corrected” in order that the motor can maintain the profile without exceeding T_SOLL. As a consequence of the correction to velocity, FNC_OUT, P_FNC is also corrected. See "FNC_OUT - Register 117", page 43

If a relative positioning is performed by repetitively adding the relative position to P_SOLL, it is only a matter of time before the contents of P_SOLL will be ”out of range”. See "P_SOLL - Register 3", page 18.

Assume for example that P_SOLL = 0 and P_FNC = 0. A relative positioning of 100000 encoder counts is carried out by adding 100000 to P_SOLL. The function generator will perform this positioning by generating a velocity profile. Once the positioning is complete, P_FNC = 1600000, since the units for P_FNC are 1/16 encoder count. This relative positioning operation can thus be carried out a maximum number of 670 times before P_SOLL will be “out of range”.

Alternatively the relative positioning can be performed by *subtracting* 1600000 from P_FNC. Here too, the function generator will perform a relative positioning. Once the positioning is complete, P_FNC = 0. This relative positioning can be repeated indefinitely. The disadvantage of this method of relative positioning is that no information is maintained about the absolute position that results from repeated relative positioning.

See "P_FNC - Register 8", page 22, and the RELPOSPSOLL and RELPOSPFNC bits under "ERR_STAT - Register 35", page 31.
See "T_SOLL - Register 7", page 21, "FNC_OUT - Register 117", page 43, and "Speed Regulator", page 64.

4.1 Function blocks (Profile Generation)

4.1.3 Gear Follow Function block

This function block is designed to achieve the least possible follow error in relation to incoming pulses. On the other hand, no account is taken in the event that if the incoming pulse-train suddenly stops, the motor will positionally overshoot before stopping at the correct position. See “Gear Position Function block” , page 62.

The function block generates a velocity profile with the following parameters:

- Measured velocity at the external pulse generator or external encoder. See "V_EXT - Register 120", page 44
- GEARF1, GEARF2: Gear factor = GEARF1 / GEARF2. See "GEARF1 - Register 14", page 25.
- V_SOLL: Maximum velocity for compensating for “follow error”.
- A_SOLL: Maximum allowable acceleration / deceleration for compensating for follow error.

The following registers are updated:

- Velocity profile is written to register: FNC_OUT. See "FNC_OUT - Register 117", page 43.
- P_FNC. See below.

If the motor is overloaded mechanically, such that the torque necessary to maintain a given velocity profile exceeds T_SOLL, the velocity profile is ”corrected” to enable the motor to maintain the profile without exceeding T_SOLL. Corrections are accumulated in the register: P_FNC. The reason that the motor is overloaded mechanically will typically be that the ”acceleration” at an external pulse generator is too large.

When the overload condition ceases, the function block will transfer the contents of P_FNC to the velocity profile, so that compensation is made for the follow error. However, the maximum allowable velocity, V_SOLL, will not be exceeded.

The function can, in principle, be described as follows:

- FNC_OUT = V_EXT * GEARF1 / GEARF2
- G_FNC = G_FNC - FNC_OUT
- V_NEW = -G_FNC
- Limit V_NEW to +/- V_SOLL.
- DV = V_NEW – V_OLD
- Limit DV to +/- A_SOLL.
- FNC_OUT = V_OLD + DV
- ACC_FLAG = FNC_OUT > V_OLD.
- DEC_FLAG = FNC_OUT < V_OLD
- V_OLD = FNC_OUT
- G_FNC = G_FNC + FNC_OUT.

See "FNC_OUT - Register 117", page 43, "GEARF1 - Register 14", page 25, "GEARF2 - Register 15", page 25, "V_SOLL - Register 5", page 19, "A_SOLL - Register 6", page 20, and "CNTRL_BITS - Register 36", page 32.

4.1 Function blocks (Profile Generation)

4.1.4 Torque function block

This function block generates a *torque* as follows:

- The analogue input is converted such that +/- 10V corresponds to +/- T_SOLL.

The following registers are updated:

- Torque is written to register VF_OUT. See "VF_OUT - Register 121", page 44.

When this function block is used, the velocity loops are passive. Therefore the torque is written to the velocity loop's output register, VF_OUT.

The sampling frequency for the analogue input is 7812 Hz.

4.1.5 AV function block

This function block generates a velocity profile with the following parameters:

- The analogue input is converted such that +/- 10V corresponds to +/- V_SOLL. See "ANINP - Register 122", page 44.
- A_SOLL: Maximum acceleration / deceleration in order to achieve velocity.

The following registers are updated:

- Velocity profile is written to register: FNC_OUT.
  See "FNC_OUT - Register 117", page 43.

4.1.6 AVZ function block

This function block is identical to the AV block, with the exception that a deadband of +/- 0.6 Volt is inserted in the conversion of the analogue voltage to motor velocity (w).

The conversion is carried out as follows:

```
IF ( ANINP > 64 ) THEN
      w = ( ANINP - 64 ) * 0.0010416 * V_SOLL
IF ( -64 < ANINP <= 64 ) THEN
      w = 0
IF ( ANINP <= -64 ) THEN
      w = ( ANINP + 64 ) * 0.0010416 * V_SOLL
```

See "ANINP - Register 122", page 44, "AV function block", page 58.

4.1 Function blocks (Profile Generation)

4.1.7 AVG function block

This function block generates a velocity profile with the following parameters:

- GEARF1, GEARF2: Gear factor = GEARF1 / GEARF2.
  See "GEARF1 - Register 14", page 25.
- The velocity at an external pulse generator or external encoder is measured and multiplied by the gear factor.
- The analogue input is converted such that +/- 10V corresponds to +/- 300 rpm, and is added. See "ANINP - Register 122", page 44.
- V_SOLL: Maximum velocity for compensating for follow error.
- A_SOLL: Maximum allowable acceleration / deceleration for compensating for follow error.

The following registers are updated:

- Velocity profile is written to register: FNC_OUT.
  See "FNC_OUT - Register 117", page 43.

4.1.8 Stop function block

This function block generates a velocity profile as follows:

- The motor decelerates in accordance with register A_SOLL to 0 RPM. See "A_SOLL - Register 6", page 20, and "ACC_EMERG - Register 32", page 30.
- If the motor rotation is less than 300 RPM, the mode is automatically switched to Init_mode.

This function is used automatically if a change of mode to INIT_MODE occurs from any mode of operation in which it is assumed that the motor is rotating. Such a switch of operating mode can be due to a motor command or due to a motor error condition. See "Velocity Mode", page 68, "Position Mode", page 69, "Gear Position Mode (GP_MODE, MODE = 3)", page 69, "Analog Velocity Mode (AV_MODE, MODE = 5)", page 72, and "Analogue Velocity Gear Mode (AVG_MODE, MODE = 6)", page 72.

The following registers are updated:

- Velocity profile is written to register: FNC_OUT.
  See "FNC_OUT - Register 117", page 43.

4.1 Function blocks (Profile Generation)

4.1.9 Analogue Gear Function block.

This function block generates a velocity profile with the following parameters:

- GEARF1, GEARF2: Gear factor = GEARF1 / GEARF2.
  See "GEARF1 - Register 14", page 25.
- The analogue input is converted to an analogue gear factor such that +/- 10V corresponds to 1.15 .. 0.87. See "ANINP - Register 122", page 44.
- The velocity at an external pulse generator or external encoder is measured and multiplied by: (gear factor * analogue gear factor). The effective gear factor can thus be adjusted using an analogue voltage. In contrast to the AVG function block, this method of synchronising the MacMotor with an external signal ensures that when the external signal's "velocity" is zero, the MacMotor will be stationary regardless of the value of the analogue voltage.
- V_SOLL: Maximum velocity for compensating for follow error.
- A_SOLL: Maximum allowable acceleration / deceleration for compensating for follow error.

The following registers are updated:

- Velocity profile is written to register: FNC_OUT.
  See "FNC_OUT - Register 117", page 43.

4.1.10 Analogue 2 Position function block

This function block controls the positioning of the motor depending on the voltage applied to the analogue input. The analogue input is used as a logical input in which a programmatic hysteresis is built in, such that a change from low to high state is registered when the voltage exceeds 5.4 V, and a change from high to low state is registered when the voltage is less than 4.4 V.

Positioning can be carried out in three ways:

• Absolute positioning: This method of positioning is used when RELPSOLL=0 and RELPFNC=0. See "CNTRL_BITS - Register 36" , page 32. On changing from low to high analog input voltage, P_SOLL = P2 is set. On changing from high to low analogue input voltage, P_SOLL = P1 is set.
• Relative positioning using P_SOLL: This method of positioning is used when RELPSOLL=1 and RELPFNC=0. See "CNTRL_BITS - Register 36" , page 32. On changing from low to high analogue input voltage, P_SOLL = P_SOLL +P2 is set. On changing from high to low analogue input voltage, P_SOLL = P_SOLL + P1 is set.
• Relative positioning using P_FNC: This method of positioning is used when RELPSOLL=0 and RELPFNC=1. On changing from low to high analogue input voltage P_FNC = P_FNC - P2 is set. On changing from high to low analogue input voltage, P_FNC = P_FNC - P1 is set.

When P_SOLL or P_FNC are calculated, the rest of the block's function is executed in principally the same way as the Position function block. See "Position function block" , page 55.

4.1 Function blocks (Profile Generation)

4.1.11 Analogue Position function block

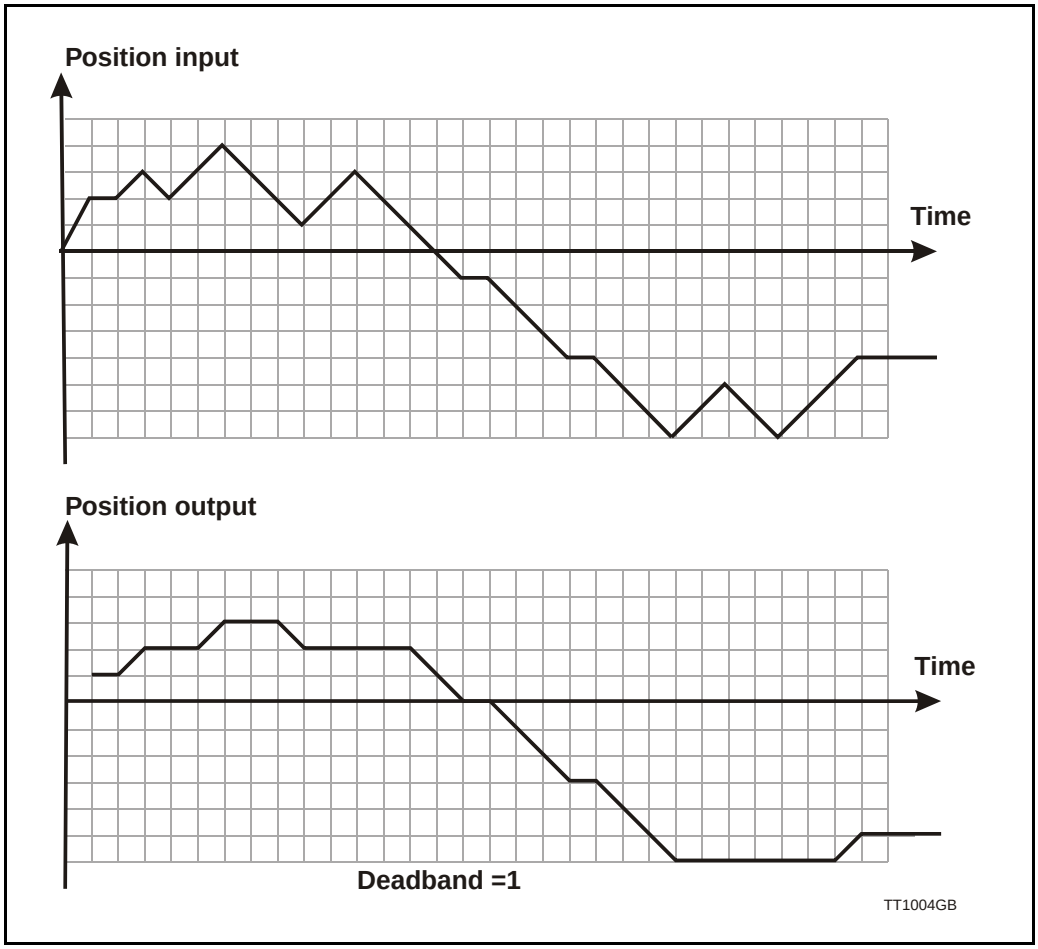
This function block controls the positioning of the motor depending on the voltage applied to the analogue input. This function block can only be used for absolute positioning.

The block's function can be described as follows:

```
P_NEW = ANINP * P2 / 16 + P1.
IF ( P_NEW < P_SOLL ) THEN
      P_NEW = P_NEW + P3
      IF ( P_NEW < P_SOLL ) THEN P_SOLL = P_NEW
ELSE
      P_NEW = P_NEW - P3
      IF ( P_NEW > P_SOLL ) THEN P_SOLL = P_NEW
```

- P2 is used for gearing between the analogue voltage and digital position.
- P1 is used for the position offset.
- P3 is used to specify an analogue deadband. The purpose of this deadband is on the basis of an input function — to produce an output function in which the dynamic variation is minimal, at the same time as ensuring the output function's deviation from the input function is maximum the width of the deadband. The function is illustrated below.

4.1 Function blocks (Profile Generation)



When P_SOLL is calculated, execution of the rest of the block's function is in principle the same as that of the Position function block. See "Position function block" , page 55.

When selecting gearing between the analogue voltage and digital position, and when choosing the bandwidth of the analogue deadband, account must be taken of the fact that the analogue signal contains noise. Similarly, some quantisation noise in the AD-conversion must be anticipated. If this noise is transformed to significant noise in P_SOLL, there is a risk that the motor will be thermally overloaded and an error condition will occur.

4.1.12 Gear Position Function block

This function block is designed to stop the motor at the correct position without overshooting, in relation to incoming pulses. On the other hand, no account is made of achieving the least possible follow error during positioning. See "Gear Follow Function block" , page 57.

The function block generates a velocity profile with the following parameters:

- Measured velocity at the external pulse generator or external encoder. See "V_EXT - Register 120", page 44
- GEARF1, GEARF2: Gear factor = GEARF1 / GEARF2.
  See "GEARF1 - Register 14", page 25.

- V_SOLL: Maximum velocity for compensating for “follow error”.
- A_SOLL: Maximum allowable acceleration / deceleration for compensating for follow error.

The following registers are updated:

- Velocity profile is written to register: FNC_OUT.
  See "FNC_OUT - Register 117", page 43.
- P_FNC. See below.

If the motor is overloaded mechanically, such that the torque necessary to maintain a given velocity profile exceeds T_SOLL, the velocity profile is ”corrected” to enable the motor to maintain the profile without exceeding T_SOLL. Corrections are accumulated in the register: P_FNC. The reason that the motor is overloaded mechanically will typically be that the ”acceleration” at an external pulse generator is too large.

When the overload condition ceases, the function block will transfer the contents of P_FNC to the velocity profile, so that compensation is made for the follow error. However, the maximum allowable velocity, V_SOLL, will not be exceeded.

The function can, in principle, be described as follows:

- FNC_OUT = V_EXT * GEARF1 / GEARF2
- G_FNC = G_FNC – FNC_OUT
- T = (|V_OLD|) / A_SOLL, (calculated deceleration time).
- FNC_OUT = -2 * G_FNC / T.
- Limit FNC_OUT to +/- V_SOLL.
- DV = V_SOLL – V_OLD
- Limit DV to +/- A_SOLL.
- FNC_OUT = V_OLD + DV
- ACC_FLAG = FNC_OUT > V_OLD.
- DEC_FLAG = FNC_OUT < V_OLD
- V_OLD = FNC_OUT
- G_FNC = G_FNC + FNC_OUT.

See "FNC_OUT - Register 117", page 43, "GEARF1 - Register 14", page 25, "GEARF2 - Register 15", page 25, "V_SOLL - Register 5", page 19, "A_SOLL - Register 6", page 20, and "CNTRL_BITS - Register 36", page 32.

4.2 Speed Regulator

4.2.1 Speed regulator

The input to the speed regulator is always output from a function block, FNC_OUT. See "FNC_OUT - Register 117", page 43. See also illustration below on this page.

The value of FNC_OUT passes through the FF filter. This filter can be used to fine-tune the velocity profile generated by the function block. In addition, the filter is used to compensate for the dynamic characteristics of the subsequent regulation loop so that the motor's "follow error" is very small. The output of the FF-filter is written to the register FF_OUT. See "FF_OUT - Register 118", page 43, and "KFF3 - Register 89", page 40.

Encoder-counts are sampled and V_IST are calculated as follows:

$$V\_IST = \frac{z - 1}{z} * \text{Encoder counts}$$

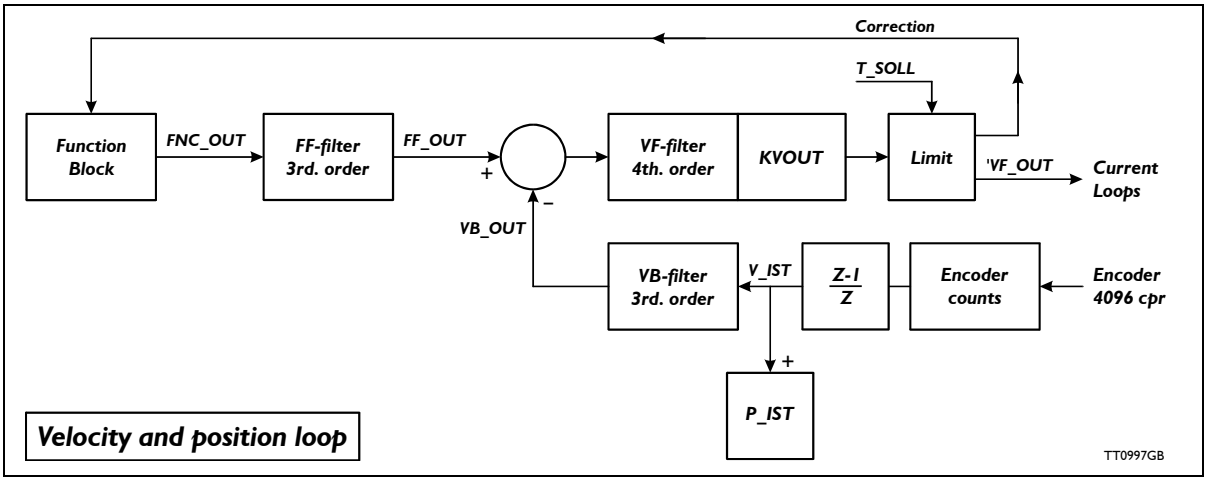
V_IST is added to P_IST.

V_IST passes through the VB filter. See "VB_OUT - Register 119", page 43. The output of the VB-filter is written to VB_OUT.

The velocity error, (FF_OUT - VB_OUT), passes through the VF filter. See "KVFX4 - Register 93", page 40. The output of this filter is multiplied by KVOUT, and the result (torque) is written to register VF_OUT. See "KVOUT - Register 13", page 24, and "VF_OUT - Register 121", page 44.

If the absolute value of VF_OUT > T_SOLL, (maximum allowable torque), VF_OUT is limited. In those modes of operation where function correction is active (P_MODE and G_MODE), a calculation is made of the speed the function block should have generated (FNC_OUT) in order that the allowable torque is not exceeded. Those registers that are affected by this correction (P_FNC, FNCERR, FNC_OUT, etc.) are corrected. See "P_FNC - Register 8", page 22, and "FNCERR", page 28.

The calculated torque, VF_OUT, is input for the current regulator. See "Current Regulator", page 65.



4.3 Current Regulator

4.3.1 Current regulator

The input to the current regulator is always the contents of the register, VF_OUT, which is the output of:

- The speed regulator. See "Speed Regulator", page 64.
- The torque function block. See "Torque function block", page 58.

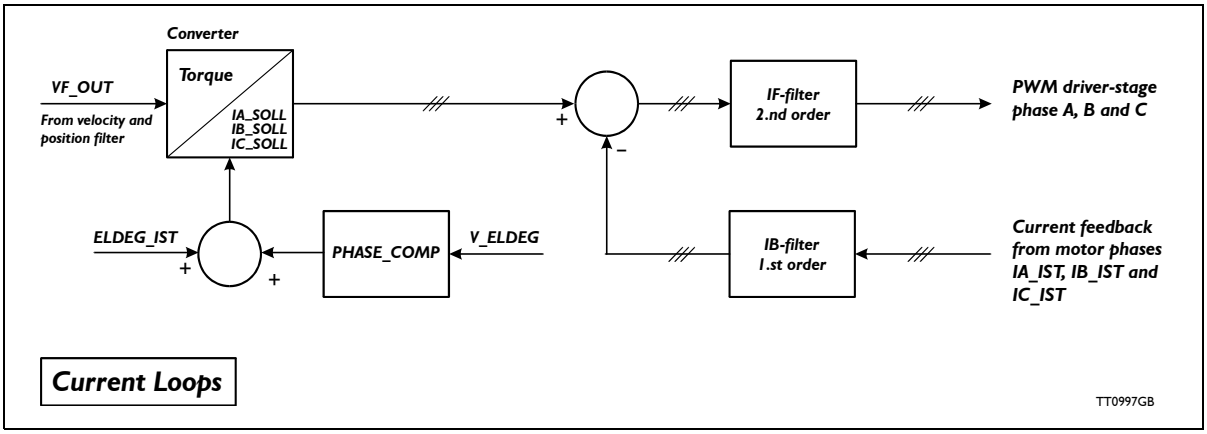
On the basis of VF_OUT, (torque), the amplitude and commutation angle, (I_NOM,PHI_SOLL), for the stator field are calculated. See "ELDEGN_OFFSET - Register 124", page 45, "ELDEGP_OFFSET - Register 125", page 45, "PHASE_COMP - Register 126", page 45, "PHI_SOLL - Register 132", page 47, "ELDEG_IST - Register 143", page 48, and "V_ELDEG - Register 144", page 49.

The three phase currents are calculated as follows:

- IA_SOLL = I_NOM * AMPLITUDE * sin(PHI_SOLL)
- IB_SOLL = I_NOM * AMPLITUDE * sin(PHI_SOLL + 120 electr. degrees)
- IC_SOLL = I_NOM * AMPLITUDE * sin(PHI_SOLL + 240 electr. degrees)

See "IA_SOLL - Register 133", page 47 and "AMPLITUDE - Register 127", page 46.

The three phase currents are regulated as illustrated below. See also "KIFX2 - Register 106", page 41, and "KIB1 - Register 110", page 42, regarding the current filters.



4.4 Modes

The following modes are implemented:

- Init mode, INIT_MODE(MODE_REG = 0)
- Velocity mode, V_MODE(MODE_REG = 1)
- Position mode, P_MODE(MODE_REG = 2)
- Gear Position mode, GP_MODE(MODE_REG = 3)
- Analog torque mode, AT_MODE(MODE_REG = 4)
- Analog Velocity mode, AV_MODE(MODE_REG = 5)
- Gear mode + Analog Velocity, AVG_MODE(MODE_REG = 6)
- Manual mode, MANI_MODE(MODE_REG = 7)
- TESTU_MODE(MODE_REG = 8)
- TESTA_MODE(MODE_REG = 9)
- BREAK_MODE(MODE_REG = 10)
- STOP_MODE(MODE_REG = 11)
- Home1 mode, HOME1_MODE(MODE_REG = 12)
- Home2 mode, HOME2_MODE(MODE_REG = 13)
- Home3 mode, HOME3_MODE(MODE_REG = 14)
- Programming mode, SAFE_MODE(MODE_REG = 15)
- Analogue Velocity mode with deadband, AVZ_MODE. See “Analogue Velocity Mode with Deadband, AVZ_MODE (MODE = 16)” , page 79.
- Velocity with Analogue Torque, VAT_MODE. See “Velocity Analogue Torque Mode, VAT_MODE (MODE = 17).” , page 79.
- Analogue Gear, AG_MODE. See “Analogue Gear Mode, AG_MODE (MODE = 18)” , page 79.
- Coiling, COIL_MODE. See “Coil Mode, COIL_MODE (MODE = 19)” , page 79.
- Analogue 2 position mode, A2POS_MODE. See “Analogue 2 Position mode, A2POS_MODE (MODE = 20)” , page 82. (MODE = 20).
- Analogue Position mode. See “Analogue Position mode, APOS_MODE (MODE = 21)” , page 82. (MODE = 21).
- TestKI mode, TESTKI_MODE. (MODE = 22).
- TestTQ mode. (MODE = 23).
- Gear Follow mode, GF_MODE. (MODE = 24).

In the various modes of operation, the function blocks, the speed regulator and current-regulator are configured in different manners.

The following sub-sections describe the configuration used to drive the motor in each of the different modes of operation, and specify which registers and flags are updated.

The following registers and bits are updated in all modes of operation:

- I2T: Calculated coil temperature of the motor. See "I2T - Register 16", page 26 and "I²T", page 86.
- UIT: Calculated load of internal power dump. See "UIT - Register 18", page 26 and "UIT", page 86.
- I2T_ERR: Bit that indicates coil temperature overload. See "ERR_STAT - Register 35", page 31 and "I²T", page 86.
- UIT_ERR: Bit that indicates overload of internal power dump. See "ERR_STAT - Register 35", page 31 and "UIT", page 86.
- P_LIM_ERR: Bit that indicates that a software position limit has been exceeded. See "ERR_STAT - Register 35", page 31 and "Software end-of-travel limits", page 86.

4.4 Modes

4.4.1 Init Mode

In this mode of operation, the same constant voltage is applied to the motor coils, i.e. the motor coils are short-circuited. This has the effect that when the motor is rotated mechanically, the short-circuit current will brake the motor.

In Init mode the configuration is as follows:

- Function block: None.
- Speed Regulator: Passive.
- Current Regulator: Passive. The same constant voltage is applied to all motor phases, i.e. the motor is short-circuited. See "IA_OFFSET - Register 140", page 48.

The following registers and flags are updated:

- P_IST. See "P_IST - Register 10", page 22.
- V_IST. See "V_IST - Register 12", page 23.
- IN_POS bit = 0. See "ERR_STAT - Register 35", page 31, "INPOSWIN - Register 33", page 30, and "INPOSCNT - Register 34", page 30.
- FLWERR = 0. See "ERR_STAT - Register 35", page 31, "FLWERR - Register 20", page 27, and "Follow error", page 87.
- FNCERR = 0. See "ERR_STAT - Register 35", page 31, "FNCERR", page 28, and "Function Error", page 87.
- IA, IB and IC_OFFSET are adjusted. See "IA_OFFSET - Register 140", page 48.

If a switch to Init Mode is made from a mode in which the MAC motor must be assumed to be moving (Velocity Mode, Position Mode, etc.), the MAC motor will automatically first switch to Stop Mode, before switching to Init Mode. See "STOP_MODE (MODE = 11)", page 75.

A switch from Init-mode to another mode can only be made if no error bits are set. See "ERR_STAT - Register 35", page 31.

4.4.2 Velocity Mode

In Velocity Mode the configuration is as follows:

- Function blocks: Velocity function block. See "Velocity function block", page 54.
- Speed Regulator: Active.
- Current Regulator: Active.
- Correction of function block: Passive.

This implies that in Velocity Mode the parameters are interpreted as follows:

- V_SOLL: Required velocity.
- A_SOLL: Maximum allowable acceleration / deceleration to achieve required velocity.
- T_SOLL: Maximum allowable torque to achieve required velocity.

The following registers and flags are updated in Velocity Mode:

- P_IST. See "P_IST - Register 10", page 22.
- V_IST. See "V_IST - Register 12", page 23.
- FLWERR. Accumulated value of velocity error. See "ERR_STAT - Register 35", page 31, "FLWERR - Register 20", page 27, and "Follow error", page 87.
- FNCERR. See "ERR_STAT - Register 35", page 31, "FNCERR", page 28, and "Function Error", page 87.
- IN_POS bit in ERRSTAT is set as a function of the value of the INPOSCNT register:
  If INPOSCNT $>$ 0 IN_POS is set if $|$ FLWERR $|$ $<$ INPOSWIN.
  If INPOSCNT $=$ 0 IN_POS is set if $|$ V_IST $|$ $<$ INPOSWIN.
  See "ERR_STAT - Register 35", page 31, "INPOSWIN - Register 33", page 30, "INPOSCNT - Register 34", page 30, "V_IST - Register 12", page 23 and "Function blocks (Profile Generation)", page 54.
- FLW_ERR. See "ERR_STAT - Register 35", page 31.
- ACC_FLAG. See "ERR_STAT - Register 35", page 31 and "Velocity function block", page 54.
- DEC_FLAG. See "ERR_STAT - Register 35", page 31 and "Velocity function block", page 54.

In this mode of operation, a change to INIT_MODE always occurs via STOP_MODE in order to protect the motor.

4.4.3 Position Mode

In Position Mode the configuration is as follows:

- Function block: Position function block. See "Position function block", page 55.
- Speed Regulator: Active. See "Speed Regulator", page 64.
- Current Regulator: Active. See "Current Regulator", page 65.
- Correction of function block: Active. See "Position function block", page 55.

This implies that in Position Mode the parameters are interpreted as follows:

- P_SOLL: Required absolute position.
- V_SOLL: Maximum velocity to achieve required position.
- A_SOLL: Maximum allowable acceleration / deceleration to achieve required position.
- T_SOLL: Maximum allowable torque to achieve required position.

The following registers and flags are updated in Position Mode:

- P_IST. See "P_IST - Register 10", page 22.
- V_IST. See "V_IST - Register 12", page 23.
- FLWERR: Accumulated value of velocity error. See "ERR_STAT - Register 35", page 31, "FLWERR - Register 20", page 27, and "Follow error", page 87.
- FNCERR: Accumulated value of correction of function block. See "ERR_STAT - Register 35", page 31, "FNCERR", page 28, and "Function Error", page 87.
- IN_POS bit in ERRSTAT is set if $|\ P\_SOLL - P\_FNC/16 + FLWERR\ | < INPOSWIN$. See "ERR_STAT - Register 35", page 31, "INPOSWIN - Register 33", page 30, "INPOSCNT - Register 34", page 30, "P_SOLL - Register 3", page 18, "P_FNC - Register 8", page 22, "FLWERR - Register 20", page 27, and "Position function block", page 55.
- FNC_ERR. See "ERR_STAT - Register 35", page 31.
- FLW_ERR. See "ERR_STAT - Register 35", page 31.
- ACC_FLAG. See "ERR_STAT - Register 35", page 31 and "Position function block", page 55.
- DEC_FLAG. See "ERR_STAT - Register 35", page 31 and "Position function block", page 55.

In this mode of operation, a change to INIT_MODE always occurs via STOP_MODE in order to protect the motor.

4.4.4 Gear Position Mode (GP_MODE, MODE = 3)

In this mode of operation the configuration is as follows:

- Function block: Gear Position function block. See “Gear Position Function block” , page 62.
- Speed Regulator: Active. See "Speed Regulator", page 64.
- Current Regulator: Active. See "Current Regulator", page 65.
- Correction of function block: Active. See "Gear Follow Function block", page 57.

This implies that in this mode of operation the primary parameters are interpreted as follows:

- GEARF1, GEARF2: Gear factor = GEARF1 / GEARF2. Reversal of the direction of rotation can be made by changing the sign of GEARF1. Nominally, in GEAR_MODE the motor is driven with 4096 pulses per revolution. If it is required to drive the motor at 500 pulses per revolution, GEARF1 / GEARF2 is specified as: 4096 / 500, or more optimally: 1024 / 125. See "GEARF1 - Register 14", page 25.
- V_SOLL: Maximum velocity for compensating for follow error. See "V_SOLL - Register 5", page 19.
- A_SOLL: Maximum allowable acceleration / deceleration for compensating for follow error. See "A_SOLL - Register 6", page 20.
- T_SOLL: Maximum allowable torque to achieve required position. See "T_SOLL - Register 7", page 21.

The following primary registers and flags are updated in Gear Mode:

- P_IST. See "P_IST - Register 10", page 22.
- V_IST. See "V_IST - Register 12", page 23.
- FLWERR: Accumulated value of velocity error. See "ERR_STAT - Register 35", page 31, "FLWERR - Register 20", page 27, and "Follow error", page 87.
- FNCERR: Accumulated value of correction of function block. See "ERR_STAT - Register 35", page 31, "FNCERR", page 28, and "Function Error", page 87.
- IN_POS bit in ERRSTAT is set if | P_FNC | < INPOSWIN. See "ERR_STAT - Register 35", page 31, "INPOSWIN - Register 33", page 30, "INPOSCNT - Register 34", page 30, and "Function blocks (Profile Generation)", page 54.
- FNC_ERR. See "ERR_STAT - Register 35", page 31.
- FLW_ERR. See "ERR_STAT - Register 35", page 31.

In this mode of operation, a change to INIT_MODE always occurs via STOP_MODE in order to protect the motor. See “Analogue Gear Mode, AG_MODE (MODE = 18)” , page 79.

4.4.5 Analog Torque Mode (AT_MODE, MODE = 4)

In this mode of operation the configuration is as follows:

- Function block: AT-function block. See "Torque function block", page 58.
- Speed Regulator: Passive.
- Current Regulator: Active. See "Current Regulator", page 65.
- Correction of function block: Passive

This implies that in this mode of operation the primary parameters are interpreted as follows:

- T_SOLL: Maximum torque, (+/- 10V is converted to +/- T_SOLL). See "ANINP - Register 122", page 44.

The following primary registers and flags are updated in Analog Torque-mode:

- P_IST. See "P_IST - Register 10", page 22.
- V_IST. See "V_IST - Register 12", page 23.
- FLWERR = 0. See "ERR_STAT - Register 35", page 31.
- FNCERR = 0. See "ERR_STAT - Register 35", page 31.
- IN_POS = 0. See "ERR_STAT - Register 35", page 31.

When a change from AT_MODE to another mode of operation occurs, in order to protect the motor the following occurs:

- Torque is set to 0.
- A wait occurs until the motor is rotating at low speed.
- Mode is switched to INIT_MODE.

It is thus only possible to switch to INIT_MODE from AT_MODE.

4.4.6 Analog Velocity Mode (AV_MODE, MODE = 5)

In this mode of operation the configuration is as follows:

- Function block: AV-function block. See "AV function block", page 58.
- Speed Regulator: Active. See "Speed Regulator", page 64.
- Current Regulator: Active. See "Current Regulator", page 65.
- Correction of function block: Passive

This implies that in this mode of operation the primary parameters are interpreted as follows:

- V_SOLL: Required maximum velocity, (+/- 10V analogue input is converted to +/- V_SOLL).
- A_SOLL: Maximum allowable acceleration/deceleration to achieve required velocity.
- T_SOLL: Maximum allowable torque to achieve required velocity.

The following primary registers and flags are updated in Analog Torque-mode:

- P_IST. See "P_IST - Register 10", page 22.
- V_IST. See "V_IST - Register 12", page 23.
- FLWERR: Accumulated value of velocity error. See "FLWERR - Register 20", page 27.
- FNCERR = 0
- IN_POS = 0. See "ERR_STAT - Register 35", page 31.

In this mode of operation, a change to INIT_MODE always occurs via STOP_MODE in order to protect the motor.

4.4.7 Analogue Velocity Gear Mode (AVG_MODE, MODE = 6)

In this mode of operation the configuration is as follows:

- Function block: AVG-function block. See "AVG function block", page 59.
- Speed Regulator: Active. See "Speed Regulator", page 64.
- Current Regulator: Active. See "Current Regulator", page 65.
- Correction of function block: Passive

This implies that in this mode of operation the primary parameters are interpreted as follows:

- GEARF1, GEARF2: Gear factor = GEARF1 / GEARF2.
  See "GEARF1 - Register 14", page 25.
- V_SOLL: Maximum velocity for compensating for follow error.
  See "V_SOLL - Register 5", page 19.
- ANINP: +/- 10V is converted to +/- 300 RPM. See "ANINP - Register 122", page 44.
- A_SOLL: Maximum allowable acceleration / deceleration for compensating for follow error. See "A_SOLL - Register 6", page 20.
- T_SOLL: Maximum allowable torque. See "T_SOLL - Register 7", page 21.

The following primary registers and flags are updated in AVG Mode:

- P_IST. See "P_IST - Register 10", page 22.
- V_IST. See "V_IST - Register 12", page 23.

- FLWERR: Accumulated value of velocity error. See "FLWERR - Register 20", page 27.
- FNCERR = 0
- FLW_ERR. Accumulated value of velocity error.
  See "ERR_STAT - Register 35", page 31.
- IN_POS = 0. See "ERR_STAT - Register 35", page 31.

In this mode of operation, a change to INIT_MODE always occurs via STOP_MODE in order to protect the motor.

For example, this mode of operation can be used in applications where two or more motors must be fundamentally operated synchronously for manufacturing of e.g. sheet material. The analogue input here is used for fine tuning the synchronisation if the material is stretched or wrinkled during manufacture.

4.4.8 Manual Mode (MANI_MODE, MODE = 7)

In this mode of operation the configuration is as follows:

- Function block: None.
- Speed Regulator: Passive.
- Current Regulator: Active.
- Correction of function block: Passive

Primary parameters:

- MAN_I_NOM: Nominal peak current. See "MAN_I_NOM - Register 128", page 46.
- MAN_ALPHA: Electrical commutation angle.
  See "MAN_ALPHA - Register 129", page 46

The following primary registers and flags are updated in MANI_MODE:

- P_IST: Measured position.
- V_IST: Measured velocity.
- FLWERR = 0.
- FNCERR = 0
- FLW_ERR = 0
- IN_POS = 0

This mode of operation is used for manual control of the stator field. It is used, for example, in conjunction with adjustment of ELDEG_OFFSET. See "ELDEGP_OFFSET - Register 125", page 45, and "ELDEG_IST - Register 143", page 48.

4.4.9 TESTU_MODE (MODE = 8)

In this mode of operation the configuration is as follows:

- Function block: None.
- Speed Regulator: Passive.
- Current Regulator: Passive.
- Correction of function block: Passive

Parameters:

- UMEAS: Applied voltage step during measurement.
  See "UMEAS - Register 130", page 47.

This mode of operation performs the following function:

- SAMPLE1 = 145. (Sample UA_VAL to the sample buffer.
  See "SAMPLE1 - Register 112", page 43)
- SAMPLE2 = 139. (Sample IC_IST to the sample buffer).
- REWINDBIT = 1. (Sample from start of buffer.
  See "CNTRL_BITS - Register 36", page 32)
- RECINNERBIT = 1. (Sampling frequency = 7812 Hz.
  See "CNTRL_BITS - Register 36", page 32)
- RECORDBIT = 1. (Start sampling. See "CNTRL_BITS - Register 36", page 32).
- UA_VAL = UMEAS, UC_VAL = -UMEAS.
- Sampling is carried out for 11 sampling periods.
- Switch to INIT_MODE.

Using the measurements in the sample buffer, the transfer function of the motor coils can be identified.

4.4.10 TESTA_MODE (MODE = 9)

In this mode of operation the configuration is as follows:

- Function block: None.
- Speed Regulator: Passive.
- Current Regulator: Active.
- Correction of function block: Passive

Parameters:

- T_SOLL: Applied torque during measurement.

This mode of operation performs the following function:

- SAMPLE1 = 7. (Sample T_SOLL to the sample buffer.
  See "SAMPLE1 - Register 112", page 43)
- SAMPLE2 = 12. (Sample V_IST to the sample buffer).
- REWINDBIT = 1. (Sample from start of buffer.
  See "CNTRL_BITS - Register 36", page 32)
- RECINNERBIT = 0. (Sampling frequency = 520.8 Hz.
  See "CNTRL_BITS - Register 36", page 32)
- RECORDBIT = 1. (Start sampling. See "CNTRL_BITS - Register 36", page 32).
- Sampling is carried out for 11 sampling periods.
- Switch to BREAK_MODE. See "BREAK_MODE (MODE = 10)", page 75.

Using the measurements in the sample buffer, the transfer function for motor dynamics can be identified.

4.4.11 BREAK_MODE (MODE = 10)

A switch to this mode of operation occurs on completion of TESTA_MODE. See "TESTA_MODE (MODE = 9)", page 74.

In this mode of operation, the sign of T_SOLL is changed for braking the motor after acceleration test.

After braking, a mode switch is made to INIT_MODE.

4.4.12 STOP_MODE (MODE = 11)

In this mode of operation the configuration is as follows:

- Function block: Stop function block. See "Stop function block", page 59.
- Speed Regulator: Active. See "Speed Regulator", page 64.
- Current Regulator: Active. See "Current Regulator", page 65.
- Correction of function block: Passive

Parameters:

- A_SOLL: Maximum allowable acceleration / deceleration.
  See "A_SOLL - Register 6", page 20, and "ACC_EMERG - Register 32", page 30.

The following primary registers and flags are updated in STOP_MODE:

- P_IST: Measured position.
- V_IST: Measured velocity.

This mode of operation is used as a transitional mode when a switch of mode is made from an operating mode in which it is assumed that the motor is rotating to INIT_MODE, in which the motor is short-circuited. See "Init Mode", page 67. A switch of operating mode to INIT_MODE may occur as a result of a command or as a resulted of a detected error condition.

If the motor is short-circuited at high rates of rotation, the short-circuit current may damage the motor. This mode of operation has therefore been introduced to facilitate controlled braking of the motor. See "ERR_STAT - Register 35", page 31 and "ACC_EMERG - Register 32", page 30.

This mode automatically switches to INIT_MODE when the motor velocity is measured to be less than 300 RPM.

4.4.13 HOME1_MODE (MODE = 12)

In this mode of operation the configuration is as follows:

- Function block: Velocity function block. See "Velocity function block", page 54.
- Speed Regulator: Active. See "Speed Regulator", page 64.
- Current Regulator: Active. See "Current Regulator", page 65.
- Correction of function block: Passive

Parameters:

- V_HOME: Velocity used in conjunction with zero-point search.
  See "V_HOME - Register 40", page 34.
- T_HOME: Torque limit for detecting mechanical limit.
  See "T_HOME - Register 41", page 35
- P_HOME: Defined positional value for zero-point.
  See "P_HOME - Register 38", page 34.
- A_SOLL: Maximum acceleration during search.

The following primary registers and flags are updated in HOME1_MODE-mode:

- P_IST. See "P_IST - Register 10", page 22.
- V_IST. See "V_IST - Register 12", page 23.
- FLWERR: Accumulated value of velocity error. See "FLWERR - Register 20", page 27.
- FNCERR = 0
- FLW_ERR. See "ERR_STAT - Register 35", page 31.
- IN_POS = 0. See "ERR_STAT - Register 35", page 31.

This mode of operation performs the following function:

- V_SOLL = V_HOME. See "V_HOME - Register 40", page 34.
- T_SOLL = 1.5 * T_HOME. (Allow higher peak torque than detection limit). See "T_HOME - Register 41", page 35.
- Reset a detection counter. (Not accessible to user).
- If the applied torque (VF_OUT) exceeds T_HOME during operation, the detection counter is incremented. See "VF_OUT - Register 121", page 44.
- Zero position is defined when the detection counter = 50.

At the zero position, the following occurs:

- P_IST = P_HOME. See "P_IST - Register 10", page 22.
- P_FNC = P_HOME * 16. See "P_FNC - Register 8", page 22.
- P_SOLL = 0
- T_SOLL = original value.
- MODE_REG = STARTMODE. See "MODE_REG - Register 2", page 17, and "STARTMODE - Register 37", page 34.

4.4.14 HOME2_MODE (MODE = 13)

In this mode of operation the configuration is as follows:

- Function block: Velocity function block. See "Velocity function block", page 54.
- Speed Regulator: Active. See "Speed Regulator", page 64.
- Current Regulator: Active. See "Current Regulator", page 65.
- Correction of function block: Passive

Parameters:

- V_HOME: Velocity used in conjunction with zero-point search.
  See "V_HOME - Register 40", page 34.
- T_SOLL: Maximum torque used during search. See "T_SOLL - Register 7", page 21.
- P_HOME: Defined positional value for zero point. See "P_HOME - Register 38", page 34.

- A_SOLL: Maximum acceleration during search. See "A_SOLL - Register 6", page 20.
- T_HOME: The sign of T_HOME indicates the active level of the home sensor. See "T_HOME - Register 41", page 35.

The following registers and flags are updated in HOME2_MODE:

- P_IST. See "P_IST - Register 10", page 22.
- V_IST. See "V_IST - Register 12", page 23.
- FLWERR: Accumulated value of velocity error. See "FLWERR - Register 20", page 27.
- FNCERR = 0
- FLW_ERR. See "ERR_STAT - Register 35", page 31.
- IN_POS = 0. See "ERR_STAT - Register 35", page 31.

This mode of operation performs the following function:

- V_SOLL = -V_HOME, (Move away from home sensor). See "V_HOME - Register 40", page 34.
- If home sensor is passive, set V_SOLL = V_HOME. (Move towards home sensor).
- Zero position is defined when the home sensor is activated.

At the zero position, the following occurs:

- P_IST = P_HOME. See "P_IST - Register 10", page 22.
- P_FNC = P_HOME * 16. See "P_FNC - Register 8", page 22.
- P_SOLL = 0
- MODE_REG = STARTMODE. See "MODE_REG - Register 2", page 17, and "STARTMODE - Register 37", page 34.

4.4.15 HOME3_MODE (MODE = 14)

In this mode of operation the configuration is as follows:

- Function block: Velocity function block. See "Velocity function block", page 54.
- Speed Regulator: Active. See "Speed Regulator", page 64.
- Current Regulator: Active. See "Current Regulator", page 65.
- Correction of function block: Passive

Parameters:

- V_HOME: Velocity used in conjunction with zero-point search. See "V_HOME - Register 40", page 34.
- T_SOLL: Maximum torque used during search. See "T_SOLL - Register 7", page 21.
- P_HOME: Defined positional value for zero position. See "P_HOME - Register 38", page 34.
- A_SOLL: Maximum acceleration during search. See "A_SOLL - Register 6", page 20.
- T_HOME: The sign of T_HOME indicates the active level of the home sensor. See "T_HOME - Register 41", page 35.

The following registers and flags are updated in HOME3_MODE:

- P_IST. See "P_IST - Register 10", page 22.
- V_IST. See "V_IST - Register 12", page 23.
- FLWERR: Accumulated value of velocity error. See "FLWERR - Register 20", page 27.
- FNCERR = 0

- FLW_ERR. See "ERR_STAT - Register 35", page 31.
- IN_POS = 0. See "ERR_STAT - Register 35", page 31.

This mode of operation performs the following function:

- V_SOLL = -V_HOME, (Move away from home sensor). See "V_HOME - Register 40", page 34.
- If home sensor is passive, set V_SOLL = V_HOME. (Move towards home sensor).
- If home sensor is active, set V_SOLL = -V_HOME / 64 + 1. (Move slowly away from sensor without stopping).
- The zero position is defined when the home sensor is deactivated.

At the zero position, the following occurs:

- P_IST = P_HOME. See "P_IST - Register 10", page 22.
- P_FNC = P_HOME * 16. See "P_FNC - Register 8", page 22.
- P_SOLL = 0
- MODE_REG = STARTMODE. See "MODE_REG - Register 2", page 17, and "STARTMODE - Register 37", page 34.

4.4.16 SAFE_MODE (MODE = 15)

Functionally, this mode of operation is identical to INIT_MODE, (the motor is passive).

Various expansion modules can be installed in the MAC motor, and via the FASTMAC / FLEXMAC protocol, these modules can control various parameters. However, the MAC motor cannot be brought out of SAFE_MODE using the FASTMAC / FLEXMAC protocol, only using the MACtalk protocol.

In the event that software in an installed expansion module is faulty, so that erroneous commands control the motor incorrectly or control is lost, the motor can be brought under control by switching it to SAFE_MODE using the MACtalk protocol. The motor cannot be brought out of this mode of operation from a faulty module.

Once the faulty software has been corrected, the corrected program can then be transferred to the expansion module, and the motor then brought out of SAFE_MODE via the MACtalk protocol.

Note that the MAC motor cannot be switched to Safe Mode by writing the value 15 to the MODE_REG register. Similarly the MAC motor cannot be switched from Safe Mode to another mode by writing to MODE_REG. See "Description of MacRegIO Windows", page 11 and "MODE_REG - Register 2", page 17.

In Safe Mode, the following special values can be written to the mode register:

- 100h = 256: Default parameter set-up will be read from firmware and saved in flash-memory as userparameters. (Motor will be reset).
- 101h = 257: Default parameter set-up will be read from firmware, but is not saved in flash.
- 102h = 258: Last saved userparameters will be read from flash

See "MODE_REG - Register 2", page 17.

4.4.17 Analogue Velocity Mode with Deadband, AVZ_MODE (MODE = 16)

This mode corresponds to Analogue Velocity mode, with the exception that a dead-band is introduced in reading the analogue voltage:

- If | AIN | < 0.625 Volts then AIN = 0 Volts
- If AIN > 0.625 Volts then AIN = (AIN - 0.625) * 1.0625 Volts
- If AIN < -0.625 Volts then AIN = (AIN + 0.625) * 1.0625 Volts

See “Analog Velocity Mode (AV_MODE, MODE = 5)” , page 72.

4.4.18 Velocity Analogue Torque Mode, VAT_MODE (MODE = 17).

This mode corresponds to Velocity mode, with the exception that torque limiting is controlled by the analogue input voltage. The torque limit, (T), is calculated as follows:

T = abs (ANINP * T_SOLL / 1024)

See "Velocity Mode", page 68, "ANINP - Register 122", page 44 and "T_SOLL - Register 7", page 21.

4.4.19 Analogue Gear Mode, AG_MODE (MODE = 18)

This mode corresponds to Gear Mode, with the exception that the effective gear ratio can be adjusted using the analogue input voltage. See “Analogue Gear Function block.” , page 60. See “Gear Position Mode (GP_MODE, MODE = 3)” , page 69.

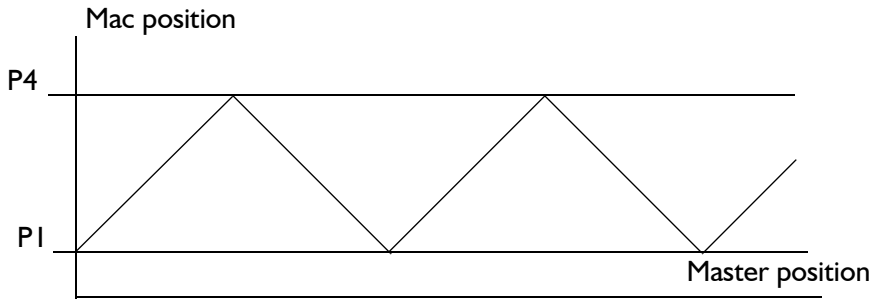
4.4.20 Coil Mode, COIL_MODE (MODE = 19)

The purpose of this mode is to facilitate spooling of wire onto a coil.

An encoder which is mechanically connected to the spindle produces a signal to the MAC motor, which must traverse for correct feeding of the coil between the spool’s layers. Essentially this mode corresponds to Gear Mode, with the extra function that the sign of the gear factor is changed at thread positions. The required increase (mm/rev.), is set via the gear factor = GEARF1 / GEARF2, as in Gear Mode. See "Gear Position Mode (GP_MODE, MODE = 3)", page 69, and "Gear Position Function block", page 62. From the external encodersignal, the value of V_EXT-register is calculated as in Gear Mode, **but in addition, the value of register V8 is added.** See "V_EXT - Register 120", page 44 and "V8 - Register 72", page 38.

For normal operation, the value of V8 must be set = 0, but when adjusting the position profile, it is not necessary having the spindle rotating. This rotation can be simulated by setting V8 to some positive or negative value.

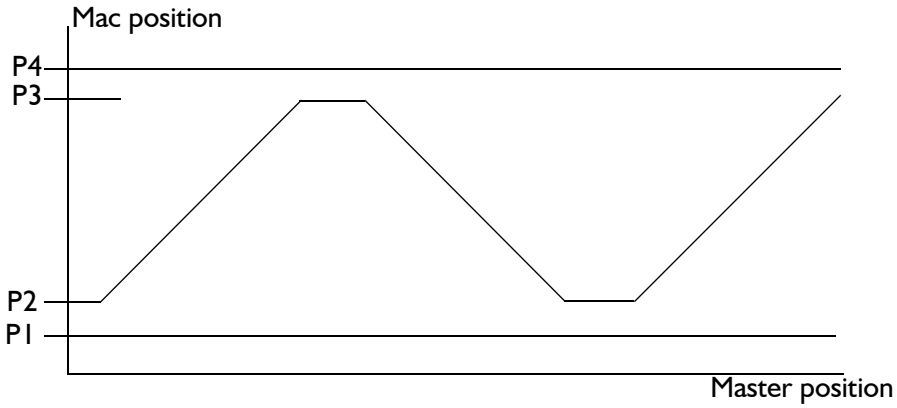
When the spool is rotated at a constant rate, the position profile shown in figure 4.4.20.A is produced:



As illustrated, the thread positions are specified by the values of P1 and P4. See “P1 - Register 49” , page 38.

If the diameter of the wire being coiled is large compared to the width of the coil (few windings per layer), it may be appropriate to stop the traverse movement while a winding is coiled onto the previous winding, during the switch of the direction of coiling.

This halt in the traverse movement is carried out by limiting the position profile, as shown in figure 4.4.20.B:



As illustrated, the stop positions are given by the values of registers P2 and P3.

At the start of a new coil operation, the start position and starting direction must be specified in order to make the coiling process repeatable. The start position is specified in register P5, and the starting direction specified in register P6.

The rules for setting up registers P1 .. P6 are:

- P1 <= P2 <= P3 <= P4
- P1 <= P5 <= P4
- P6 = +1 or -1

A typical set-up for registers P1 .. P6 is therefore:

P1 = 10000	Left thread position
P2 = 12000	Left stop position
P3 = 58000	Right stop position
P4 = 60000	Right thread position
P5 = 12000	Start at left side
P6 = 1	Start in right direction

Positioning at the start position is activated by setting the analogue input voltage, (AIN), high. When AIN is set low, the MacMotor will resume the normal coil function.

The MacMotor can be set up to perform an initiating homing at power up. When using some sensor homing mode, the homing function uses the analogue input to sense the home position and the coiling function uses the signal to sense the start position command. The two signals can be coupled in parallel if the home position sensor is activated away from normal working area, (between P1 and P4).

Note: Having completed the power-up homing, an initial start position command must be given before starting the first coil process.

4.4.21 Analogue 2 Position mode, A2POS_MODE (MODE = 20)

In Analogue 2 Position Mode the configuration is as follows:

- Function block: Analogue 2 Position function block. See “Analogue 2 Position function block” , page 60.
- Speed Regulator: Active. See "Speed Regulator", page 64.
- Current Regulator: Active. See "Current Regulator", page 65.
- Correction of function block: Active. See "Position function block", page 55.

This implies that in Analogue 2 Position Mode, the parameters are interpreted as follows:

- P1,P2: Required positioning, dependent on value of analogue input.
- V_SOLL: Maximum velocity to achieve required position.
- A_SOLL: Maximum allowable acceleration / deceleration to achieve required position.
- T_SOLL: Maximum allowable torque to achieve required position.

The following registers and flags are updated in Analogue 2 Position Mode:

- P_IST. See "P_IST - Register 10", page 22.
- V_IST. See "V_IST - Register 12", page 23.
- FLWERR: Accumulated value of velocity error. See "ERR_STAT - Register 35", page 31, "FLWERR - Register 20", page 27, and "Follow error", page 87.
- FNCERR: Accumulated value of correction of function block. See "ERR_STAT - Register 35", page 31, "FNCERR", page 28, and "Function Error", page 87.
- IN_POS bit in ERRSTAT is set if $| P\_SOLL - P\_FNC/16 + FLWERR | <$ INPOSWIN. See "ERR_STAT - Register 35", page 31, "INPOSWIN - Register 33", page 30, "INPOSCNT - Register 34", page 30, "P_SOLL - Register 3", page 18, "P_FNC - Register 8", page 22, "FLWERR - Register 20", page 27, and "Position function block", page 55.
- FNC_ERR. See "ERR_STAT - Register 35", page 31.
- FLW_ERR. See "ERR_STAT - Register 35", page 31.
- ACC_FLAG. See "ERR_STAT - Register 35", page 31 and "Position function block", page 55.
- DEC_FLAG. See "ERR_STAT - Register 35", page 31 and "Position function block", page 55.

In this mode of operation, a change to INIT_MODE always occurs via STOP_MODE in order to protect the motor.

4.4.22 Analogue Position mode, APOS_MODE (MODE = 21)

In Analogue Position Mode the configuration is as follows:

- Function block: Analogue Position function block. See “Analogue Position function block” , page 61.
- Speed Regulator: Active. See "Speed Regulator", page 64.
- Current Regulator: Active. See "Current Regulator", page 65.
- Correction of function block: Active. See "Position function block", page 55.

This implies that in Analogue Position Mode the parameters are interpreted as follows:

- P1,P2: Required positioning, dependent on value of analogue input.
- V_SOLL: Maximum velocity to achieve required position.
- A_SOLL: Maximum allowable acceleration / deceleration to achieve required position.
- T_SOLL: Maximum allowable torque to achieve required position.

The following registers and flags are updated in Analogue Position Mode:

- P_IST. See "P_IST - Register 10", page 22.
- V_IST. See "V_IST - Register 12", page 23.
- FLWERR: Accumulated value of velocity error. See "ERR_STAT - Register 35", page 31, "FLWERR - Register 20", page 27, and "Follow error", page 87.
- FNCERR: Accumulated value of correction of function block. See "ERR_STAT - Register 35", page 31, "FNCERR", page 28, and "Function Error", page 87.
- IN_POS bit in ERRSTAT is set if $|\ P\_SOLL - P\_FNC/16 + FLWERR\ | <$ INPOSWIN. See "ERR_STAT - Register 35", page 31, "INPOSWIN - Register 33", page 30, "INPOSCNT - Register 34", page 30, "P_SOLL - Register 3", page 18, "P_FNC - Register 8", page 22, "FLWERR - Register 20", page 27, and "Position function block", page 55.
- FNC_ERR. See "ERR_STAT - Register 35", page 31.
- FLW_ERR. See "ERR_STAT - Register 35", page 31.
- ACC_FLAG. See "ERR_STAT - Register 35", page 31 and "Position function block", page 55.
- DEC_FLAG. See "ERR_STAT - Register 35", page 31 and "Position function block", page 55.

In this mode of operation, a change to INIT_MODE always occurs via STOP_MODE in order to protect the motor.

4.4.23 TestKI mode, TESTKI_MODE (MODE = 22)

This mode is used for calculating the amplification in measurement circuitry for the motor phase currents. See “KIA - Register 148” , page 49.

4.4.24 TestTQ mode, TESTTQ_MODE (MODE = 23)

This mode is used for measurement of frictional torque. For test purposes only.

4.4.25 Gear Follow Mode, GF_MODE (MODE = 24)

In this mode of operation the configuration is as follows:

- Function block: Gear Follow function block. See "Gear Follow Function block", page 57.
- Speed Regulator: Active. See "Speed Regulator", page 64.
- Current Regulator: Active. See "Current Regulator", page 65.

- Correction of function block: Active. See "Gear Follow Function block", page 57.

This implies that in this mode of operation the primary parameters are interpreted as follows:

- GEARF1, GEARF2: Gear factor = GEARF1 / GEARF2. Reversal of the direction of rotation can be made by changing the sign of GEARF1. Nominally, in GEAR_MODE the motor is driven with 4096 pulses per revolution. If it is required to drive the motor at 500 pulses per revolution, GEARF1 / GEARF2 is specified as: 4096 / 500, or more optimally: 1024 / 125. See "GEARF1 - Register 14", page 25.
- V_SOLL: Maximum velocity for compensating for follow error. See "V_SOLL - Register 5", page 19.
- A_SOLL: Maximum allowable acceleration / deceleration for compensating for follow error. See "A_SOLL - Register 6", page 20.
- T_SOLL: Maximum allowable torque to achieve required position. See "T_SOLL - Register 7", page 21.

The following primary registers and flags are updated in Gear Mode:

- P_IST. See "P_IST - Register 10", page 22.
- V_IST. See "V_IST - Register 12", page 23.
- FLWERR: Accumulated value of velocity error. See "ERR_STAT - Register 35", page 31, "FLWERR - Register 20", page 27, and "Follow error", page 87.
- FNCERR: Accumulated value of correction of function block. See "ERR_STAT - Register 35", page 31, "FNCERR", page 28, and "Function Error", page 87.
- IN_POS bit in ERRSTAT is set if | P_FNC | < INPOSWIN. See "ERR_STAT - Register 35", page 31, "INPOSWIN - Register 33", page 30, "INPOSCNT - Register 34", page 30, and "Function blocks (Profile Generation)", page 54.
- FNC_ERR. See "ERR_STAT - Register 35", page 31.
- FLW_ERR. See "ERR_STAT - Register 35", page 31.

In this mode of operation, a change to INIT_MODE always occurs via STOP_MODE in order to protect the motor. See “Analogue Gear Mode, AG_MODE (MODE = 18)” , page 79.

4.5 Monitoring Functions

The motor software contains 6 monitoring functions, which are always active:

- Monitoring of the motor's coil temperature (calculated). See "I²T", page 86.
- Monitoring of the internal "power dump" temperature, (calculated). See "UIT", page 86.
- Monitoring of the software end-of-travel limits. See "Software end-of-travel limits", page 86.
- Monitoring of the "Follow error". See "Follow error", page 87.
- Monitoring of Function error. See "Function Error", page 87.
- Monitoring of undervoltage condition in the supply voltage. See "Undervoltage detection", page 87.

4.5 Monitoring Functions

4.5.1 I²T

The motor's coil temperature is calculated on the basis of the sum of the squared phase currents (current heat loss). The value of the current heat loss is accumulated in register I2T. If the value I2T exceeds the value of the I2TMAX parameter, the I2T_ERR bit in ERR-STAT is set, and a switch of mode is automatically made to INIT_MODE. See "I2T - Register 16", page 26, "I2TLIM - Register 17", page 26, and "ERR_STAT - Register 35", page 31.

WARNING: Iron losses in the rotor are not included in the calculation. These losses will be significant if the motor is driven at a greater rate of revolution than that specified for the supply voltage used.

4.5.2 UIT

The motor includes an internal "power dump". When the motor brakes, e.g. during deceleration, the motor produces regenerative energy back to the supply. The supply voltage may therefore increase to a level that can cause damage to circuitry unless a power dump is used to sink the regenerative energy. This energy heats the power dump components.

The temperature of the power dump's components is calculated on the basis of the measured regenerative energy to the power dump and the value is accumulated in the UIT register. If the value of UIT exceeds the value of the UITMAX parameter, the UIT_ERR bit is set, and the mode of operation is automatically switched to INIT_MODE via STOP_MODE. See "UIT - Register 18", page 26, "UITLIM - Register 19", page 26, and "STOP_MODE (MODE = 11)", page 75.

If this error occurs frequently, it is recommended that the user mounts an external power dump, or adjusts the acceleration / deceleration. See "A_SOLL - Register 6", page 20.

4.5.3 Software end-of-travel limits

The P_IST register, which indicates the motor's instantaneous position, is updated in all modes of operation. See "P_IST - Register 10", page 22.

If the motor's allowable position range is limited, monitoring of the motor position can be activated. This monitoring is active if register MIN_P_IST <> 0 or register MAX_P_IST <> 0. See "P_IST - Register 10", page 22, and "MIN_P_IST - Register 28", page 29.

When active, the monitoring function will set PLIM_ERR (1) if:

- P_IST < MIN_P_IST.
- P_IST > MAX_P_IST.

See "ERR_STAT - Register 35", page 31. At the same time, the motor will change to an error state and switch mode to INIT_MODE via STOP_MODE. See "STOP_MODE (MODE = 11)", page 75.

4.5.4 Follow error

The FLWERR register is updated in several modes of operation. See "FLWERR - Register 20", page 27. FLWERR is calculated by accumulating the instantaneous velocity error in these modes of operation.

If monitoring of this follow error is required, it can be activated. Follow error monitoring is active if register FLWERRMAX > 0. See "FLWERRMAX - Register 22", page 27.

When active, the monitoring function will set FLW_ERR (1) if:

- Absolute value (FLW_ERR) > FLWERRMAX.

See "ERR_STAT - Register 35", page 31. At the same time, the motor will change to an error state and switch mode to INIT_MODE via STOP_MODE. See "STOP_MODE (MODE = 11)", page 75.

4.5.5 Function Error

The FNCERR register is updated in several modes of operation. See "FNCERR", page 28. FNCERR is calculated by accumulating corrections of a function block.

If monitoring of these corrections is required, the function error monitoring can be activated. Monitoring of the function error is active if register FNCERRMAX > 0. See "FNCERRMAX", page 28.

When active, the monitoring function will set FNC_ERR (1) if:

- Absolute value (FNC_ERR) > FNCERRMAX.

See "ERR_STAT - Register 35", page 31. At the same time, the motor will change to an error state and switch mode to INIT_MODE via STOP_MODE. See "STOP_MODE (MODE = 11)", page 75.

4.5.6 Undervoltage detection

The supply voltage is continuously monitored, and the measured value is written to the U_SUPPLY register. See “U_SUPPLY - Register 151” , page 49. Depending on the set-up configuration, the programme can react if the measured supply voltage falls below the specified minimum voltage. See “U_MIN_SUP - Register 152” , page 49.

In general terms, the undervoltage monitoring and detection function can be described as follows:

```
IF ( U_SUPPLY < U_MIN_SUP ) THEN
        Undervoltage timer, UV_TIMER, is preset to 1 sec.
        Bit UV_DETECT = 1.
        Error bit, UV_ERR = SET_UV_ERR.
        IF ( UV_VSOLL0 = 0 ) THEN V_SOLL = 0

ELSE
        IF ( U_SUPPLY > 1.25*U_MIN_SUP ) THEN
                IF ( UV_TIMER = 0 ) THEN UV_DETECT = 0
```

See “ERR_STAT - Register 35” , page 31. See “UV_HANDLE - Register 160” , page 50.

In Init_mode the following is carried out:

```
IF ( U_SUPPLY < 1.25*U_MIN_SUP ) THEN
        Undervoltage timer, UV_TIMER, is preset to 1 sec.
        Bit UV_DETECT = 1.
ELSE
        IF ( UV_TIMER = 0 ) THEN UV_DETECT = 0
```

5 Interface Description

5.1 MacTalk Protocol

Using the MacTalk protocol, users can read from and write to all registers. See "Parameter and Data Registers", page 17.

The MacTalk protocol is defined by:

message := <readsync> <address> <startreg> <endsync> |
<readblocksync> <address> <startreg> <endsync> |
<writesync> <address> <startreg> <datalength> <datablock> <endsync> |
<readsamplebuffersync> <address> <endsync> |
<gosafemodesync> <address> <endsync> |
<goinitmodesync> <address> <endsync> |
<writeparmflashsync> <address> <endsync> |
<resetsync> <address> <endsync> |
<accept>

readsync := <50h> <50h> <50h>
readblocksync := <51h> <51h> <51h>
writesync := <52h> <52h> <52h>
readsamplebuffersync := <53h> <53h> <53h>
gosafemodesync := <54h> <54h> <54h>
goinitmodesync := <55h> <55h> <55h>
writeparmflashsync := <56h> <56h> <56h>
resetsync := <57h> <57h> <57h>

endsync := <0AAh> <0AAh>

address := <unitaddr> <!unitaddr>(See "MYADDR - Register 156", page 50)
unitaddr := <masteraddr> | <macaddr> | <broadcastaddr>
masteraddr := 0
macaddr := 01h .. 0FEh
broadcastaddr := 0FFh

! := not operator

startreg := <regnr> <!regnr>
regnr := 01h .. 0FFh (See "Parameter and Data Registers", page 17)

datalength := <dl> <!dl>
dl := number of dataword in datablock

datablock := <datablock> <dataword>
dataword := <databyte> <!databyte>

accept := 11h,11h,11h

Example:

Message = 50h,50h,50h,13h,ECh,07h,F8h,AAh,AAh:
From master to unit 13h: Retransmit contents of parameter number 7.

Unit 13h has no knowledge whether this parameter is in integer or longint format. Therefore the parameter is always retransmitted as if it is longint format. If it is assumed that parameter number 7 is in integer format with the value 4711h, and it is assumed that the value of parameter number 8 has the value 0000h, the following will be retransmitted:

52h,52h,52h,00h,FFh,04h,FBh,11h,EEh,47h,B8h,00h,FFh,00h,FFh,AAh,AAh.

Note that communication is made in big-endian format.

Example:

Message = 52h,52h,52h,10h,EFh,08h,F7h,02h,FDh,22h,DDh,33h,CCh,AAh,AAh.
From master to unit 10h: Overwrite parameter number 08h with the value 3322h.

Accept is retransmitted by the unit by: 11h,11h,11h

Example:

Message = 54h,54h,54h,17h,E8h,AAh,AAh:
From master to unit 17h: Go to safe mode.

Accept is retransmitted by the unit by: 11h,11h,11h

Note that after a writeparmflashsync and a resetsync, no accept is retransmitted.

5.2 FastMAC / FlexMAC

The user interface can be configured to communicate using the FastMAC / FlexMAC-protocol. See "CNTRL_BITS - Register 36", page 32.

The FastMAC-protocol is designed to achieve rapid communication by limiting all commands and responses to 1 byte. On the rather hand, FastMAC requires that the motor is controlled in *Register Mode*. The contents of register values cannot be changed under FastMAC-protocol. See sections "P_REG_P - Register 43", page 36 to "Z4 - Register 88", page 39.

The FlexMAC-protocol is designed to provide more flexible communication, at the expense of communication speed. Using FlexMAC, all parameter values and register values can be changed.
Depending on the actual requirements for speed or flexibility, it is possible to switch between the two protocols ”on the fly”.

5.2.1 FastMAC

5.2.1.1 Registers

The MAC motor contains 32 register for controlling the motor in register mode:

- 8 position registers: P[1 .. 8]. See "P1 - Register 49", page 38, and "P_REG_P - Register 43", page 36.
- 8 velocity registers: V[1 .. 8]. See "V1 - Register 65", page 38, and "V_REG_P - Register 44", page 36.
- 4 acceleration registers: A[1 .. 4]. See "A1 - Register 73", page 39 and "A_REG_P - Register 45", page 36.
- 4 torque registers: T[1 .. 4]. See "T1 - Register 77", page 39, and "T_REG_P - Register 46", page 37.
- 4 load registers: L[1 .. 4]. See "L1 - Register 81", page 39, and "L_REG_P - Register 47", page 37.
- 4 ”In_position_registers”: Z[1 .. 4]. See "Z1 - Register 85", page 39, and "Z_REG_P - Register 48", page 37.

Within each of the 6 groups, a register can be activated, so that a register set can be combined, e.g.: P6,V4, A3, T2, L1, Z1.

5.2.1.2 Modes

Under FastMAC the MAC motor can be commanded to 4 modes:

- **Init-mode:** The motor is short-circuited by PWM-outputs. In this mode a register set can be combined before start of the motor. See "Init Mode", page 67.
- **Velocity-mode:** In this mode the register set is interpreted as follows: (See "Velocity Mode", page 68)
  Activated V-register as required velocity.
  Activated A-register as max. acceleration/deceleration.
  Activated T-register as max. allowable torque.
  Activated L-register as load factor.
- **Position-mode:** In this mode the register set is interpreted as follows: (See "Position Mode", page 69)
  Activated P-register as required absolute/relative position.
  Activated V-register as max. velocity during positioning.
  Activated A-register as max. acceleration/deceleration during positioning.

Activated T-register as max. allowable torque.
Activated L-register as load factor.
Activated Z-register as tolerance for detection that the motor is in position.

- **Command_mode:** In this mode, the MAC motor will continue to operate under the previously used mode. The last activated register combination is used and cannot be arbitrarily changed in Command-mode. However four commands are available to point to register sets P1, V1, A1, T1, L1, Z1 to P4, V4, A4, T4, L4, Z4. In Command_mode the 5 register address bits are used instead to specify 32 different commands.

5.2.1.3 Toggle bit

To act as a separator to differentiate commands, FastMAC uses a toggle bit. This bit must be switched for each new command. See "CNTRL_BITS - Register 36", page 32.

5.2.1.4 Format

Commands are sent as a single byte using the following format:

Bit 7:
Toggle bit, which is switched with each new command. At the cost of some communication speed, it is possible to select that a command must be received identically twice before the command is accepted. See "CNTRL_BITS - Register 36", page 32, SAFEMAC-BIT.

Bits 6,5:
Mode bits, that indicate the mode of the Mac motor:

- 00: Init-mode.
- 01: Velocity-mode.
- 10: Position-mode.
- 11: Command-mode.

Bits 4 ... 0:
The significance of these bits depends on the mode of the Mac motor:

Init-mode, Velocity-mode, Position-mode:

In these modes, bits 4 ... 0 are interpreted as the register-groups and register-number to be activated.

- 00XXX: P-register is activated. Bit 2 ... 0 indicates register number 1 ... 8.
- 01XXX: V-register is activated. Bit 2 .. 0 indicates register number 1 ... 8.
- 100XX: A-register is activated. Bit 1 ... 0 indicates register number 1 ... 4.
- 101XX: T-register is activated. Bit 1 ... 0 indicates register number 1 ... 4.
- 110XX: L-register is activated. Bit 1 ... 0 indicates register number 1 .. 4.
- 111XX: Z-register is activated. Bit 1 ... 0 indicates register number 1 .. 4.

Note: If a P-register is activated, the IN_POS-flag = 0 is set automatically. See "ERR_STAT - Register 35", page 31, and "Position Mode", page 69.

Command-mode:

In this mode, bits 4 ... 0 are interpreted as a command number (0 ... 31)

- 00H: NOP, (No operation).
- 01H: Reset error.
- 02H: P_SOLL = 0. IN_POS = 0. See "ERR_STAT - Register 35", page 31, and "Position Mode", page 69.
- 03H: P_IST = 0.
- 04H: P_FNC = 0. (step turntable). IN_POS = 0. See "P_FNC - Register 8", page 22, "Position function block", page 55, "ERR_STAT - Register 35", page 31, and "Position Mode", page 69.
- 05H: V_SOLL = 0. See "V_SOLL - Register 5", page 19.
- 06H: T_SOLL = 0. See "T_SOLL - Register 7", page 21.
- 07H: IN_POS-flag = 0. See "ERR_STAT - Register 35", page 31, "Velocity function block", page 54, "Position function block", page 55, and "Gear Follow Function block", page 57.
- 08H: Perform a relative positioning, specified by register P7, from *instantaneous position.* Positioning is done by: P_FNC = (FLWERR - P7) * 16. See "Position function block", page 55, "P_FNC can be used to perform relative positioning. See "Position function block", page 55. See “ERR_STAT - Register 35” , page 31., bit RELPOSPFNC.gister 8", page 22 and "FLWERR - Register 20", page 27
- 09H: Perform a relative positioning, specified by register P8, from *instantaneous position.* Positioning is done by: P_FNC = (FLWERR - P8) * 16. See "Position function block", page 55, "P_FNC can be used to perform relative positioning. See "Position function block", page 55. See “ERR_STAT - Register 35” , page 31., bit RELPOSPFNC.gister 8", page 22 and "FLWERR - Register 20", page 27
- 0AH: Use rapid communication (based on 1 byte). See "CNTRL_BITS - Register 36", page 32.
- 0BH: Use safe communication (based on 2 identical bytes). See "CNTRL_BITS - Register 36", page 32.
- 0CH: Use register set: P1, V1, A1, T1, L1, Z1. See sections "P_REG_P - Register 43", page 36 to "Z_REG_P - Register 48", page 37. IN_POS = 0. See "ERR_STAT - Register 35", page 31, and "Position Mode", page 69.
- 0DH: Use register set: P2, V2, A2, T2, L2, Z2. See sections "P_REG_P - Register 43", page 36 to "Z_REG_P - Register 48", page 37. IN_POS = 0. See "ERR_STAT - Register 35", page 31, and "Position Mode", page 69.
- 0EH: Use register set: P3, V3, A3, T3, L3, Z3. See sections "P_REG_P - Register 43", page 36 to "Z_REG_P - Register 48", page 37. IN_POS = 0. See "ERR_STAT - Register 35", page 31, and "Position Mode", page 69.
- 0FH: Use register set: P4, V4, A4, T4, L4, Z4. See sections "P_REG_P - Register 43", page 36 to "Z_REG_P - Register 48", page 37. IN_POS = 0. See "ERR_STAT - Register 35", page 31, and "Position Mode", page 69.
- 10H: Home search. See "HOME_MODE - Register 42", page 35, and sections "HOME1_MODE (MODE = 12)", page 75 to "HOME3_MODE (MODE = 14)", page 77.
- 11H: P_SOLL = P_IST + P7 - FLWERR, (move relative from current position).
- 12H: P_SOLL = P_IST + P8 - FLWERR, (move relative from current position).
- 13H: Transmit slowly. See "Synchronisation", page 97.
- 14H: Use absolute positioning using position registers. i.e. Bit RELPOSPSOLL = 0, bit RELPOSPFNC = 0. See "Position function block", page 55, "ERR_STAT - Register 35", page 31 and "P_REG_P - Register 43", page 36.
- 15H: Use relative positioning using P_SOLL-register.
  i.e. Bit RELPOSPSOLL = 1, bit RELPOSPFNC = 0. See "Position function block", page 55, "ERR_STAT - Register 35", page 31 and "P_REG_P - Register 43", page 36.
- 16H: Use relative positioning using P_FNC-register.
  i.e. Bit RELPOSPSOLL = 0, bit RELPOSPFNC = 1. See "Position function block", page 55, "ERR_STAT - Register 35", page 31 and "P_REG_P - Register 43", page 36.

- 17H: NOP
- 18H: NOP
- 19H: NOP
- 1AH: NOP
- 1BH: NOP
- 1CH: NOP
- 1DH: Transmit MYADDR. See “MYADDR - Register 156” , page 50.
- 1EH: Use FASTMAC protocol, i.e. NOP
- 1FH: Use FLEXMAC protocol

5.2.1.5 Example

The MAC motor has just been switched on and is in INIT-mode:

1. Select register set: P6, V4, A4, T2.
2. Set P_IST = 0
3. Run to P6
4. Run to P7
5. Run to P5 at velocity V3

Command:Comments

```
0E0H    ; Command_mode: NOP: synchronise toggle bit
062H    ; Set P_IST = 0
081H    ; Init_mode: Activate T2
003H    ; Init_mode: Activate A4
093H    ; Init_mode: Activate V4
05DH    ; Position_mode: Activate P6. IN_POS = 0.
        ; Wait IN_POS = 1
0DEH    ; Position_mode: Activate P7. IN_POS = 0.
        ; Wait IN_POS = 1
052H    ; Position_mode: Activate V3
0DCH    ; Position_mode: Activate P5. IN_POS = 0.
        ; Wait IN_POS = 1
```

5.2.1.6 Status

The MAC motor will repetitively transmit a status byte with the following format:

Bit 7: Toggle bit, which has the same value as the toggle bit in the last accepted command. See "CNTRL_BITS - Register 36", page 32.

Bit 6: Deceleration flag. See "ERR_STAT - Register 35", page 31, "Velocity function block", page 54, and "Position function block", page 55.

Bit 5: Acceleration flag. See "ERR_STAT - Register 35", page 31, "Velocity function block", page 54, and "Position function block", page 55.

Bit 4: In_Position_flag: This flag is updated in Position_mode. See "INPOSWIN - Register 33", page 30, "INPOSCNT - Register 34", page 30, and "ERR_STAT - Register 35", page 31.

Bit 3: Active protocol, (receive): FastMAC = 0, FlexMAC = 1. See "CNTRL_BITS - Register 36", page 32, and "FlexMAC", page 97.

Bit 2: Character received with framing error. See "Synchronisation", page 97.

Bit 1: Available.

Bit 0: Follow / Function / I2T / UIT – error: Read ERR_STAT using FlexMAC. See "ERR_STAT - Register 35", page 31.

5.2.1.7 Synchronisation

Using the Fast/FlexMAC-protocol, characters are transmitted immediately in succession. With the FastMAC-protocol these characters will typically be the same. If synchronisation is lost (incorrect bit is interpreted as the start bit by the external unit), the external unit will register this as a framing-error.

In such cases, the external unit can send the command: Transmit slowly. This has the effect that the MAC motor will insert a pause between each transmitted character so that the external unit can re-synchronise. See "Format", page 93, command-mode.

If the MAC motor receives a character with a framing error, the MAC motor will set the bit FRAME_ERR_TX. See "ERR_STAT - Register 35", page 31. In addition, the MAC motor will transmit status bytes with bit 2 set (character received with framing error). See "Status", page 96. As a consequence of this, an external device should send characters with a pause between them until synchronisation is re-established and the MAC motor sends status bytes with bit 2 reset.

5.2.2 FlexMAC

Commands in FlexMAC consist of several bytes, and a synchronisation mechanism is thus included in the form of 3 header-bytes. These header-bytes contain information about which parameter is to be written to/read from, and whether the data format used is word / longword.

Using the FlexMAC-protocol, communication can be made at 3 different levels:

1) Write to 15 common registers, (word or longword).
   Read from 15 common registers, (word or longword).

2) Write to all registers, (word or longword).
   Read from all registers, (word or longword).

3) Write to arbitrary address, (word or longword). (64 Kbyte)
   Read from arbitrary address, (word or longword). (64 Kbyte)

Syntax:

Message ::= < WriteMessage > | < ReadMessage >

WriteMessage ::= < WriteWordMessage > | < WriteLongMessage >

ReadMessage ::= < ReadWordMessage > | < ReadLongMessage >

WriteWordMessage ::= < WriteWordBlock > < WordDataBlock >

WriteLongMessage ::= < WriteLongBlock > < LongDataBlock >

ReadWordMessage ::= < ReadWordBlock >

ReadLongMessage ::= < ReadLongBlock >

WriteWordBlock ::= (< HeaderW2Q > | < HeaderW2 > < RegNmb >
< Header2W > 0 < AddressBlock >

```
WriteLongBlock ::= ( < HeaderW4Q > | < HeaderW4 > < RegNmb > |
          < Header4W > 0 < AddressBlock >

ReadWordBlock ::= < HeaderR2Q > | < HeaderR2 > < RegNmb > |
         < HeaderR2 > 0 < AddressBlock >
ReadLongBlock ::= < HeaderR4Q > | < HeaderR4 > < RegNmb > |
         < HeaderR4 > 0 < AddressBlock >

HeaderW2Q ::= < W2Q > < W2Q > < W2Q >
HeaderW2 ::= < W2 > < W2 > < W2 >
HeaderW4Q ::= < W4Q > < W4Q > < W4Q >
HeaderW4 ::= < W4 > < W4 > < W4 >
HeaderR2Q ::= < R2Q > < R2Q > < R2Q >
HeaderR2 ::= < R2 > < R2 > < R2 >
HeaderR4Q ::= < R4Q > < R4Q > < R4Q >
HeaderR4 ::= < R4 > < R4 > < R4 >

W2 ::= 060H
W2Q ::= W2 + < RegIndex >

W4 ::= 0E0H
W4Q ::= W4 + < RegIndex >

R2 ::= 070H
R2Q ::= R2 + < RegIndex >

R4 ::= 0F0H
R4Q ::= R4 + < RegIndex >

RegIndex ::= < Index > !< Index >
Index ::= 1H .. 0FH

RegNmb ::= < Nmb > !< Nmb >
Nmb ::= 1H .. 0FFH

AddressBlock ::= < WordDataBlock >

WordDataBlock ::= < ByteBlock > < ByteBlock >

ByteBlock ::= < Byte > !< Byte >

Byte ::= 0H .. 0FFH

LongDataBlock ::= ByteBlock  ByteBlock  ByteBlock  ByteBlock
```

Note: *Data Format used: Big Endian*
Note: " ! " = NOT-function

Register index tables:

Write:

1: CNTRL_BITS(word)
2: P_SOLL(longword)
3: V_SOLL(word)
4: A_SOLL(word)
5: T_SOLL(word)
6: KVOUT(word)
7: INPOSWIN(word)
8: P0(longword)
9: P1(longword)
10: V1(word)
11: V2(word)
12: A1(word)
13: A2(word)
14: L1(word)
15: L2(word)

Read:

1: CNTRL_BITS(word)
2: P_FNC(longword)
3: P_IST(longword)
4: V_IST(word)
5: V_EXT(word)
6: ANINP(word)
7: I_NOM(word)
8: VF_OUT(word)
9: I2T(word)
10: UIT(word)
11: FLWERR(word)
12: FNCERR(word)
13: ERRBIT(word)
14: Command: Use FastMAC(word)
15: Command: Use FlexMAC(word) (NOP)

Header byte format:

Bit 7: Word = 0; LongWord = 1

Bit 6,5: Always 11

Bit 4: Write = 0; Read = 1.

Bit 3..0: RegIndex.

Example:

Write: V_SOLL = 0547H

WriteIndex = 3, format = word.

Message = 063H, 063H, 063H, 047H, 0B8H, 005H, 0FAH

Or: (register number for V_SOLL = 09H):

Message = 060H, 060H, 060H, 009H, 0F6H, 047H, 0B8H, 005H, 0FAH

Or: Address for V_SOLL = 0412H

Message = 060H, 060H, 060H, 000H, 0FFH, 012H, 0EDH, 004H, 0FBH, 047H, 0B8H, 005H, 0FAH

Example:

Read P_IST:

ReadIndex = 3, format = longword

Message = 0F3H, 0F3H, 0F3H

Or: (register number for P_IST = 0EH):

Message = 0F0H, 0F0H, 0F0H, 00EH, 0F1H

Or: (Address for P_IST = 041CH):

Message = 0F0H, 0F0H, 0F0H, 000H, 0FFH, 01CH, 0E3H, 004H, 0FBH

Example:

Switch to FastMAC protocol:

Message = 07FH, 07FH, 07FH

Use of FastMAC / FlexMAC-protocol.

The MAC motor *always* uses the FastMAC-protocol to *transmit*, i.e. Status byte is transmitted repetitively, *except* when the MAC responds to a ReadMessage under FlexMAC protocol.

The MAC motor *always* uses FastMAC / FlexMAC-protocol, as commanded, to *receive*.

For each FlexMAC-message the MAC has accepted, the toggle bit will be switched in the status byte.

In the status byte under FastMAC protocol, bits 6,5 = DEC_FLAG, ACC_FLAG. These bits therefore cannot have a status = 11. Conversely, bits 6,5 in header bytes under FlexMAC are *always* set to: 11. Thus an external unit can differentiate between a status byte under FastMAC protocol and a header byte under FlexMAC-protocol when transmission is made from the MAC motor to the external unit.

The status byte contains flags that indicate under which protocol the MAC will receive a message from an external unit.

From an external unit, the protocol is switched from FastMAC to FlexMAC as follows:

- Transmit command: Use FlexMAC-protocol.
- Wait for bit 3 in status byte = 1: FlexMAC is used.
- Transmit using FlexMAC.
- During reception, an external unit differentiates between the protocol used for transmission via bits 6,5 in the status byte / header byte.

From an external unit, the protocol is switched from FlexMAC to FastMAC as follows:

- Transmit command: Use FastMAC-protocol.
- Wait for bit 3 i status byte = 0
- Transmit using FastMAC.

6 Expansion Modules

Contents for Chapter 6

6.1 MAC00-R1, R3 and R4

6.1.1 Introduction

The MAC00-R1, R3 or R4 also called the "NanoPLC" comprises:

- Six logical hardware-inputs.
- Four logical hardware-outputs
- A 16-bit timer
- A 16-bit counter
- A FastMAC/FlexMAC communication port for communication with the MACmotor, (basic PCB).
- A MACtalk communication port for communication with the user.

The NanoPLC is programmed as a sequence machine with up to 63 sequence steps. The sequencing program can be transmitted via the MACtalk communication port, and saved in FLASH-memory.

6.1.2 Sequence Machines

In a sequence machine, the control process is divided into sequence steps. At each sequence step, the process state is know, e.g. the direction of rotation of the motor. Therefore in any given sequence step, it is normally only necessary to monitor a few logical states. For example, it is not necessary to monitor an end-of-travel limit when it is known that the motor is moving away from the end-stop.

In a sequence machine, only the states and commands that are associated with the current sequence are evaluated and executed.

6.1.3 Sequence Steps

At any given sequence step, one or more conditions can be programmed for execution of one or more commands.

Programming a sequence step is done by specifying one or more sequence terms in a sequence table.

6.1.4 Format of Sequence Terms

A sequence term consists of 8 bytes in the sequence table.

A sequence term can be divided into a conditional statement and a command statement.

The conditional statement can contain a product term (a logical expression that only contains AND and NOT operators). If a condition consists of a sum-of-products term, several sequence terms must be programmed within the same sequence step in order to include an OR-operator in the condition.

The command statement may include one or a few commands. If several commands are to be executed for a given condition, several sequence steps that have the same conditional statement must be programmed within the same sequence step.

6.1.5 Conditional Statement of a Sequence Term

The conditional part of a sequence term consists of 3 bytes, which are denoted:

STATE byte.
INPUT byte
CARE byte.

STATE byte:
Bits 5 .. 0 of this byte indicate the number of the sequence step to which this sequence term belongs, i.e. the conditional statement is only evaluated as true if the number corresponds to the current sequence step.

Bit 6, COUNTER_BIT, indicates if the counter-value must be equal to 0, i.e. the conditional statement is only evaluated as true if:
COUNTER_BIT = 0 and Counter-value = 0, or
COUNTER_BIT = 1

Bit 7, TIMER_BIT, indicates if the timer-value must be equal to 0, i.e. the conditional statement is only evaluated as true if:
TIMER_BIT = 0 and Timer-value = 0, or
TIMER_BIT = 1

INPUT byte:
Bits 5 .. 0 of this byte indicate, together with the CARE byte, the status of hardware-inputs, before the conditional statement is evaluated as true. See CARE byte.

Bit 6, INPOS bit, indicates, together with the CARE byte, if the MACmotor must be "In position", before the conditional statement is evaluated as true. See CARE byte.

Bit 7, ERROR bit, indicates if the MACmotor must be "In Error", before the conditional statement is evaluated as true. The CARE byte has no influence on evaluation of this bit.

CARE byte:
Bits 6 .. 0 of this byte indicate care / don't care masks for the corresponding bits in the INPUT byte.

For example: The following condition is to be programmed:

Hardware-input[0] = 1
Hardware-input[3] = 0
INPOS bit = 1
ERROR bit = 0

Or, more formally:
IF INPUT0 AND NOT INPUT3 AND INPOS AND NOT ERROR THEN . . .

The condition is programmed as:
INPUT byte =0 1 X X 0 X X 1
CARE byte =X 1 0 0 1 0 0 1

Bit 7 of the CARE byte, CMD_FORMAT bit, indicates the format of the command statement for the sequence term. See following section.

6.1.6 Command Statement of the Sequence Term

The command statement of a sequence term consists of 5 bytes.

The command statement can be programmed in two different formats, denoted as the FastMAC-format and the FlexMAC-format.

In FastMAC-format, the command statement contains a command (one byte), which can be transmitted to the MAC-motor using the FastMAC-protocol.

In FlexMAC-format, the command statement contains a register number and register data, so that a register value can be written to the MAC-motor using the FlexMAC-protocol.

6.1.7 FastMAC Format

This format is used if the CMD_FORMAT bit = 0.

In FastMAC-format the command statement contains 3 bytes + 1 word, in all 5 bytes:

NEXT_STATE byte.
FAST_COMMAND byte
OUTPUT byte
TIMER_COUNTER word

NEXT_STATE byte:

Bits 5 .. 0 indicate the number of the sequence step to switch to if the conditional statement is evaluated as true. If bits 5 .. 0 are specified as 3Eh, this is interpreted as an illegal number and no switch is made.

Bit 6, CNT_TIME bit, indicates if the value of the TIMER_COUNTER word is a counter value (CNT_TIME bit = 1) or a timer value (CNT_TIME bit = 0). The timer or counter value is preset if the conditional statement is evaluated true and TIMER_COUNTER word > 0. If the TIMER_COUNTER word = 0, the timer- or counter-value is not updated. It is thus not possible to reset the timer or counter by setting COUNTER_TIMER word = 0.

Bit 7: Available.

FAST_COMMAND byte:

Bits 6 .. 0 specify the command to be transmitted to the MACmotor using the FastMAC protocol. See MAC User Manual, section x.x.x.

Bit 7, FAST_DIS bit, indicates if the command is to be transmitted to the MACmotor if the conditional statement is evaluated true. Transmission is disabled by specifying FAST_DIS bit = 1.

OUTPUT byte:

Bits 3 .. 0 indicate the status of hardware-outputs, if the conditional statement is evaluated as true.

Bits 5 .. 4: Available.

Bit 6, CLR_CNT bit, enables the counter to be reset, (CLR_CNT = 1).

Bit 7, CNT_DWN bit, enables the counter-value to be decremented, (CNT_DWN = 1).

TIMER_COUNTER word:

This word specifies the value to which the timer or counter is to be preset.

If the value is specified as 0, the timer / counter is not updated. To reset the counter, see OUTPUT byte. To reset the timer, the value can be specified as 1, (1 ms).

6.1.8 FlexMAC Format

This format is used if the CMD_FORMAT bit = 1.

In FlexMAC-format, the command statement contains 1 byte + 2 words, in all 5 bytes:

REG_NMB byte.
LS_DATE word
MS_DATA word

REG_NMB byte:

Bits 6 .. 0 indicate the number of the MACmotor register to be written to using the FlexMAC-protocol. See MAC User Manual, section x.x.x.

Bit 7, LONGWORD bit, indicates the register format: word / longword.

LS_DATA word:
This word specifies the least significant word to be transmitted to the MACmotor register.

MS_DATA word:
This word specifies the most significant word to be transmitted to the MACmotor register, if the format is longword.

6.1.9 Execution of the Sequence Table

Due to synchronisation, execution of the Nano-PLC sequence table is as follows:

1. Logical hardware inputs are read and written to HW_INP byte[5..0]. The INPOS bit and ERROR bit are read by a status byte, which is continuously transmitted by the MACmotor, and written to HW_INP byte[7..6].
2. All sequence terms in the sequence table are executed, that is to say the HW_INP byte is used in evaluation of the conditional statement. If the conditional statement of a sequence terms is evaluated as true, the commands specified by the command statement are written to temporary command registers. The commands are not executed instantaneously. If the conditional statement in several sequence terms is evaluated true, the temporary command registers are overwritten. If the commands conflict with one another, the command-set in the last sequence term evaluated as true will take precedence. The exception to this is transmission of the MODE_COMMAND byte using the FastMAC-format and FlexMAC-format, which is always executed instantaneously.
3. The commands in the temporary command registers are executed. Logical hardware-outputs are updated.
4. The Timer value is adjusted.
   Return to step 1.

   Re 2.: Since the Nano-PLC executes transmission of commands to the MACmotor immediately, queues can occur during transmission. To maintain synchronisation, the Nano-PLC must wait for execution of the queue, which means that execution of the sequence table is halted.

   Care should therefore be taken when using the FlexMAC-format, since this involves transmission of many bytes, with consequently long execution pauses.

   The sequence table is completed by CURRENT_STATE byte = 63, (0x3F). Thus the legal numbers for sequence steps are 0 .. 62, i.e. up to 63 sequence steps can be programmed.

7 Index

7 Index

7 Index