

**MAC50  
MAC95  
MAC140**

# **Integrated Servo Motors Technical Manual**



TT0940GB

**JVL Industri Elektronik A/S**

Copyright 1998-2002, JVL Industri Elektronik A/S. All rights reserved.  
This user manual must not be reproduced in any form without prior written permission of JVL Industri Elektronik A/S.  
JVL Industri Elektronik A/S reserves the right to make changes to information contained in this manual without prior notice.  
Similarly JVL Industri Elektronik A/S assumes no liability for printing errors or other omissions or discrepancies in this user manual.

*MotoWare is a registered trademark*

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](mailto:jvl@jvl.dk)  
Internet: <http://www.jvl.dk>

# Contents

---

<b>I</b>	<b>Introduction .....</b>	<b>5</b>
1.1	Introduction .....	6
1.2	Sampled Systems .....	7
<b>2</b>	<b>MacRegIO .....</b>	<b>9</b>
2.1	Installation .....	10
2.2	Description of MacRegIO Windows .....	11
2.3	Operation .....	13
<b>3</b>	<b>Parameter and Data Description .....</b>	<b>15</b>
3.1	Description of Data Formats .....	16
3.2	Parameter and Data Registers .....	17
<b>4</b>	<b>Program/Function Description .....</b>	<b>49</b>
4.1	Function blocks (Profile Generation) .....	50
4.2	Speed Regulator .....	55
4.3	Current Regulator .....	56
4.4	Modes .....	57
4.5	Monitoring Functions .....	68
<b>5</b>	<b>Interface Description .....</b>	<b>71</b>
5.1	MacTalk Protocol .....	72
5.2	FastMAC / FlexMAC .....	74
<b>6</b>	<b>Index.....</b>	<b>85</b>





This Technical Manual describes advanced use of the MAC motor.

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.





**MacRegIO.exe** is a program for directly writing to and reading 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.

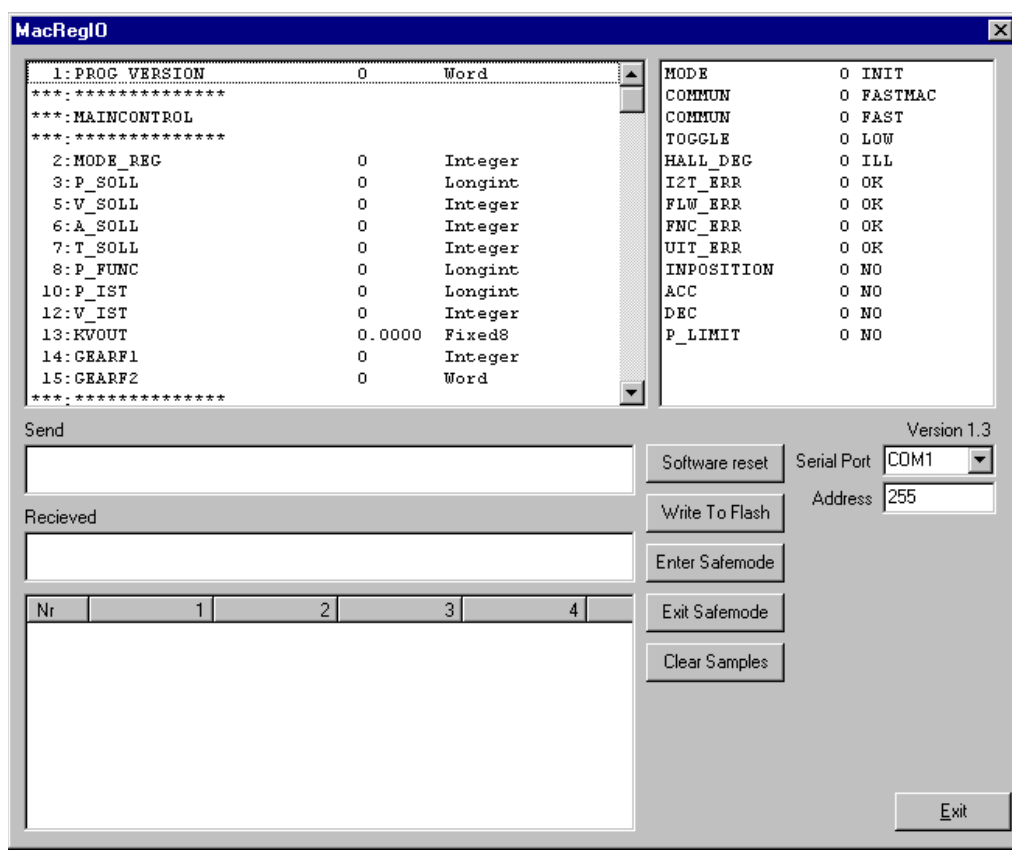
MacRegIO.exe uses two data files:

- **MacReg.dat:** This file describes the register names and register formats for various registers.
- **MacBit.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 5 buttons, as illustrated below.



### Parameter Window:

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:

This window displays the names of bits/bit groups, their numerical values and explanatory texts of the values.

### Transmit Window:

This window displays the last transmitted message, byte-by-byte, in decimal format.

### Receive Window:

This window displays the last received message, byte-by-byte, in decimal format.

### Sample Window:

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 41.

## 2.2 Description of MacRegIO Windows

---

### Software Reset Button:

This button is used to reset the MAC motor. The MAC motor can only be reset in Init Mode. See "*Init Mode*", page 57.

### 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.

### 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 67.

### Exit Safe Mode:

This button is used to exit Safe Mode.

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.

## 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 described 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 ... 156.

### 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 ... 15.

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

Value: Operation mode:

- 0 Init mode. See "*Init Mode*", page 57.
- 1 Velocity mode, V\_MODE. See "*Velocity Mode*", page 58.
- 2 Position mode, P\_MODE. See "*Position Mode*", page 58.
- 3 Gear mode, G\_MODE. See "*Gear Mode (G\_MODE, MODE = 3)*", page 59.
- 4 Analog Torque mode, AT\_MODE. See "*Analog Torque Mode (AT\_MODE, MODE = 4)*", page 60.
- 5 Analog Velocity mode, AV\_MODE. See "*Analog Velocity Mode*", page 61.
- 6 Analog Velocity Gear mode, AVG\_MODE. See "*Analog Velocity Gear mode (AVG\_MODE, MODE = 6)*", page 61.
- 7 Manual current mode, MANI\_MODE. See "*Manual Mode (MANI\_MODE, MODE = 7)*", page 62.
- 8 Voltage test mode, TESTU\_MODE. See "*TESTU\_MODE (MODE = 8)*", page 62.
- 9 Acceleration test mode, TESTA\_MODE. See "*TESTA\_MODE (MODE = 9)*", page 63.

## 3.2 Parameter and Data Registers

---

- I0 Break mode, BREAK\_MODE. See "BREAK\_MODE (MODE = 10)", page 64.
- I1 Stop mode, STOP\_MODE. See "STOP\_MODE (MODE = 11)", page 64.
- I2 Home 1 mode, HOME1\_MODE. See "HOME1\_MODE (MODE = 12)", page 64.
- I3 Home 2 mode, HOME2\_MODE. See "HOME2\_MODE (MODE = 13)", page 65.
- I4 Home 3 mode, HOME3\_MODE. See "HOME3\_MODE (MODE = 14)", page 66.
- I5 SAFE\_MODE. See "SAFE\_MODE (MODE = 15)", page 67.

### 3.2.3 P\_SOLL - Register 3

Data format: LongInt.

Range: +/-67E6.

This register indicates the absolute position to which the motor will move. See "CNTRL\_BITS - Register 36", page 31, RELPOSBIT.

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 58, "P\_IST - Register 10", page 21, and "P\_FNC - Register 8", page 21.

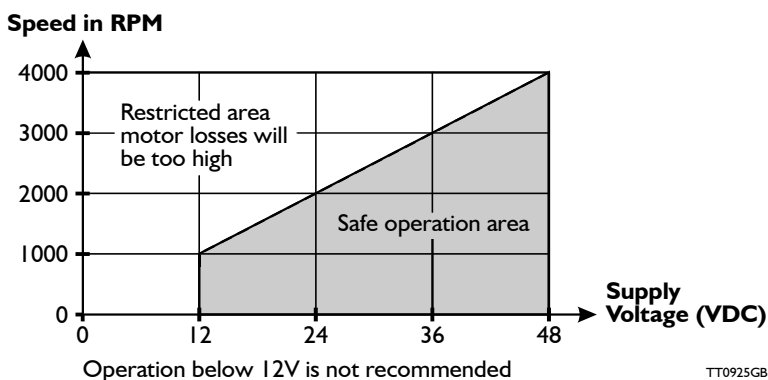
Unit: Encoder counts.

## 3.2 Parameter and Data Registers

### 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 58.
- **Position mode, P\_MODE:** V\_SOLL specifies the maximum velocity during positioning. See "Position Mode", page 58.
- **Gear mode, G\_MODE:** V\_SOLL specifies the maximum velocity during positioning. See "Gear Mode (G\_MODE, MODE = 3)", page 59.
- **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", page 61.
- **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 "Analog Velocity Gear mode (AVG\_MODE, MODE = 6)", page 61.

Unit: 0.4768 rpm.

## 3.2 Parameter and Data Registers

---

### 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<sup>2</sup>. 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 58.
- Position mode, P\_MODE. See "Position Mode", page 58.
- Gear mode, G\_MODE. See "Gear Mode (G\_MODE, MODE = 3)", page 59.
- Analog Velocity mode, AV\_MODE. See "Analog Velocity Mode", page 61.
- Analog Velocity Gear mode, AVG\_MODE.  
See "Analog Velocity Gear mode (AVG\_MODE, MODE = 6)", page 61.
- Homing mode 1 ... 3. See "HOME1\_MODE (MODE = 12)", page 64,  
"HOME2\_MODE (MODE = 13)", page 65, and  
"HOME3\_MODE (MODE = 14)", page 66.

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 58.
- Position mode, P\_MODE. See "Position Mode", page 58.
- Gear mode, G\_MODE. See "Gear Mode (G\_MODE, MODE = 3)", page 59.
- Analog Velocity mode, AV\_MODE. See "Analog Velocity Mode", page 61.
- Analog Velocity Gear mode, AVG\_MODE. See  
"Analog Velocity Gear mode (AVG\_MODE, MODE = 6)", page 61.
- TESTA\_MODE. See "TESTA\_MODE (MODE = 9)", page 63.
- HOME2\_MODE. See "HOME2\_MODE (MODE = 13)", page 65.
- HOME3\_MODE. See "HOME3\_MODE (MODE = 14)", page 66.

Full torque corresponds to a value of 1023.

## 3.2 Parameter and Data Registers

---

### 3.2.8 P\_FNC - Register 8

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 50 and "*Velocity Mode*", page 58.
- Position mode, P\_MODE. See "*Position function block*", page 50 and "*Position Mode*", page 58.
- Analog velocity mode, AV\_MODE. See "*AV function block*", page 53 and "*Analog Velocity Mode*", page 61.
- Gear mode, G\_MODE. See "*Gear function block*", page 52 and "*Gear Mode (G\_MODE, MODE = 3)*", page 59.
- Analog velocity gear mode, AVG\_MODE. See "*AVG function block*", page 54 and "*Analog Velocity Gear mode (AVG\_MODE, MODE = 6)*", page 61.

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 50.

### 3.2.10 P\_IST - Register 10

Data format: LongInt.

Range: +/-2147483647.

This register indicates the motor's current position.

Unit: Encoder counts

## 3.2 Parameter and Data Registers

---

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 68.

The register is updated in all modes of operation.

### 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  $1/520.8 \text{ sec} = 1.92 \text{ ms}$ .

Unit: 7.629 rpm.

## 3.2 Parameter and Data Registers

---

### 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 kgm<sup>2</sup>, (nominal load). The motor is loaded with an inertial load = 50E-6 kgm<sup>2</sup>.

KVOUT is calculated as:

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

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 58.
- Position mode. See "*Position Mode*", page 58.
- Gear mode. See "*Gear Mode (G\_MODE, MODE = 3)*", page 59.
- Analog Velocity mode. See "*Analog Velocity Mode*", page 61.
- Analog Velocity Gear mode. See "*Analog Velocity Gear mode (AVG\_MODE, MODE = 6)*", page 61.
- Stop mode. See "*STOP\_MODE (MODE = 11)*", page 64.
- HOME1 mode. See "*HOME1\_MODE (MODE = 12)*", page 64.
- HOME2 mode. See "*HOME2\_MODE (MODE = 13)*", page 65.
- HOME3 mode. See "*HOME3\_MODE (MODE = 14)*", page 66.

See "*Speed Regulator*", page 55.

## 3.2 Parameter and Data Registers

---

### 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 mode, G\_MODE. See "*Gear Mode (G\_MODE, MODE = 3)*", page 59.
- Analog Velocity Gear mode, AVG\_MODE. See "*Analog Velocity Gear mode (AVG\_MODE, MODE = 6)*", page 61.

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

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



## 3.2 Parameter and Data Registers

---

### 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:

$$I_a^2 + I_b^2 + I_c^2$$

where  $I_a$ ,  $I_b$  and  $I_c$  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 30.
- MODE\_REG = INIT\_MODE. See "STOP\_MODE ( MODE = 11 )", page 64 and "Init Mode", page 57.

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:

$$P_{reg} = - ( I_a \cdot U_a + I_b \cdot U_b + I_c \cdot U_c )$$

where  $I_a$ ,  $I_b$  and  $I_c$  are the measured phase currents, and  $U_a$ ,  $U_b$  and  $U_c$  are the applied voltages to the three phases.

If  $P_{reg}$  is calculated to be  $< 0$ ,  $P_{reg}$  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 30.
- MODE\_REG = INIT\_MODE. See "STOP\_MODE ( MODE = 11 )", page 64 and "Init Mode", page 57.

Register UIT is updated in all modes of operation.

## 3.2 Parameter and Data Registers

---

### 3.2.20 FLWERR - Register 20

### 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 58.
- Position mode, P\_MODE. See "*Position Mode*", page 58.
- Gear mode, G\_MODE. See "*Gear Mode (G\_MODE, MODE = 3 )*", page 59.
- Analog Velocity mode, AV\_MODE. See "*Analog Velocity Mode*", page 61.
- Analog Velocity Gear mode, AVG\_MODE. See "*Analog Velocity Gear mode ( AVG\_MODE, MODE = 6 )*", page 61.

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

V, AV, AVG\_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, G\_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 30.
- MODE\_REG = INIT\_MODE. See "*STOP\_MODE ( MODE = 11 )*", page 64 and "*Init Mode*", page 57.

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

See "*Follow error*", page 69.

## 3.2 Parameter and Data Registers

---

### 3.2.24 FNCERR

### 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 58.
- Position mode, P\_MODE. See "Position Mode", page 58.
- Gear mode, G\_MODE. See "Gear Mode (G\_MODE, MODE = 3)", page 59.

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

V\_MODE, P\_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.

It should be noted that the value of FNCERR in P\_MODE 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.

G\_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 G\_MODE, 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 30.
- MODE\_REG = INIT\_MODE. See "STOP\_MODE (MODE = 11)", page 64 and "Init Mode", page 57.

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

In P\_MODE especially, it is normally unacceptable that FNCERR deviates from 0. The specified maximum torque, T\_SOLL, and maximum acceleration, A\_SOLL, should be corrected by the user.

See "Function Error", page 69.

## 3.2 Parameter and Data Registers

---

### 3.2.28 MIN\_P\_IST - Register 28

### 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 30.
- MODE\_REG = INIT\_MODE. See "STOP\_MODE (MODE = 11)", page 64 and "Init Mode", page 57.

The monitoring function can be disabled by setting:

- MAX\_P\_IST = 0
- MIN\_P\_IST = 0

See "Software end-of-travel limits", page 68.

## 3.2 Parameter and Data Registers

---

### 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 68, "*Stop function block*", page 54, and "*STOP\_MODE ( MODE = 11 )*", page 64.

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.

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 "*Velocity Mode*", page 58, "*Position Mode*", page 58, "*Gear Mode (G\_MODE, MODE = 3 )*", page 59, and "*ERR\_STAT - Register 35*", page 30.

Recommended value for INPOSCNT: 3 – 5.

## 3.2 Parameter and Data Registers

---

### 3.2.35 ERR\_STAT - Register 35

Data format: Word.

This register contains a number of status and error bits:

- 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 50, "P\_FNC - Register 8", page 21, "P\_REG\_P - Register 43", page 34 and "Format", page 75, command-mode.
- 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 50, "P\_SOLL - Register 3", page 18, "P\_REG\_P - Register 43", page 34 and "Format", page 75, command mode.
- Bit 8: FRAME\_ERR\_TX. See "Synchronisation", page 79.
- 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 68.
- Bit 6: DEC\_FLAG: The motor is under deceleration. See "Velocity function block", page 50 and "Position function block", page 50.
- Bit 5: ACC\_FLAG: The motor is under acceleration. See "Velocity function block", page 50 and "Position function block", page 50.
- Bit 4: IN\_POS: The motor is "in position". See "Velocity Mode", page 58, "Position Mode", page 58, "Gear Mode (G\_MODE, MODE = 3)", page 59, "INPOSWIN - Register 33", page 29, and "INPOSCNT - Register 34", page 29.
- 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 25 and "UIT", page 68.
- Bit 2: FNC\_ERR: FNCERR exceeds FNCERRMAX. Mode is automatically switched to Init-mode when this error occurs. See "FNCERR", page 27 and "Function Error", page 69.
- Bit 1: FLW\_ERR: FLWERR exceeds FLWERRMAX. Mode is automatically switched to Init-mode when this error occurs. See "FLWERR - Register 20", page 26 and "Follow error", page 69.
- 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 25 and "I2T", page 68.

## 3.2 Parameter and Data Registers

### 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 43.
- Bit 11: TOGGLEBIT  
This bit indicates the status of the toggle bit in FASTMAC / FLEXMAC protocol. See "Toggle bit", page 75.
- Bit 10: SAFEMACBIT  
The FASTMAC protocol can communicate in fast mode or safe mode. If SAFEMACBIT is set ( 1 ), the protocol communicates in safe mode. See "Format", page 75
- Bit 9: FLEXMACBIT  
The FASTMAC / FLEXMAC protocol can communicate i FASTMAC mode or FLEXMAC mode. If FLEXMACBIT is set ( 1 ), communication is done in FLEXMAC mode. See "Status", page 78.
- 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 41, "Operation", page 13, "TESTU\_MODE ( MODE = 8 )", page 62, and "TESTA\_MODE ( MODE = 9 )", page 63.
- 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 41.
- Bit 6: RECÖRDBIT:  
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 41, "REC\_CNT. - Register 116", page 41, "Operation", page 13, "TESTU\_MODE ( MODE = 8 )", page 62, and "TESTA\_MODE ( MODE = 9 )", page 63.
- 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 43 and "ELDEGP\_OFFSET - Register 125", page 43.
- 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

## 3.2 Parameter and Data Registers

---

- 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 33, "HOME1\_MODE ( MODE = 12 )", page 64, "HOME2\_MODE ( MODE = 13 )", page 65 and "HOME3\_MODE ( MODE = 14 )", page 66.

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

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 64, "HOME2\_MODE ( MODE = 13 )", page 65 and "HOME3\_MODE ( MODE = 14 )", page 66.

### 3.2.40 V\_HOME - Register 40

Data format: Integer.

Recommended range: See "V\_SOLL - Register 5", page 19.



## 3.2 Parameter and Data Registers

---

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 64, "*HOME2\_MODE (MODE = 13)*", page 65 and "*HOME3\_MODE (MODE = 14)*", page 66.

The sign of V\_HOME specifies the direction of the home search.

### 3.2.41 T\_HOME - Register 41

Data format: Integer.

Range: -682 ... 682

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 1.5 times the value of T\_HOME. Since the maximum torque limit is 1023, the maximum value of T\_HOME is thus 682. See "*HOME1\_MODE (MODE = 12)*", page 64.
- 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 65 and "*HOME3\_MODE (MODE = 14)*", page 66.

### 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 75, 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 32.

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 64.
- 13: HOME2\_MODE:  
A search for the home position is made until an electrical home signal is activated. See "*HOME2\_MODE (MODE = 13)*", page 65.
- 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 66.

Once the home position has been found, the motor is started in the mode specified by STARTMODE. See "*STARTMODE - Register 37*", page 32.

## 3.2 Parameter and Data Registers

---

### 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 "PI - Register 49", page 36.

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 30.

### 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 "VI - Register 65", page 36.

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 "AI - Register 73", page 37.

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 Parameter and Data Registers

---

### 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 20.

### 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 23.

### 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 29 and "INPOSCNT - Register 34", page 29.

## 3.2 Parameter and Data Registers

---

**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 34 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 34.

## 3.2 Parameter and Data Registers

---

### 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 34.

### 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 20 and "T\_REG\_P - Register 46", page 35.

### 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 ... 6.0

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 23 and "L\_REG\_P - Register 47", page 35.

### 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 29, "INPOSCNT - Register 34", page 29 and "Z\_REG\_P - Register 48", page 35.

## 3.2 Parameter and Data Registers

---

### 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:

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

See "Speed Regulator", page 55.

### 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:

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

Note the change of sign in the transfer function.

See "Speed Regulator", page 55.

## 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:

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

See "Speed Regulator", page 55.

### 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:

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

Note the change of sign in the transfer function.

See "Current Regulator", page 56.

**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 Parameter and Data Registers

---

### 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:

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

See "*Current Regulator*", page 56.

**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 Parameter and Data Registers

---

### 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 ... 156

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 31, "TESTU\_MODE ( MODE = 8 )", page 62 and "TESTA\_MODE ( MODE = 9 )", page 63.

### 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 31.

### 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 50 and "Speed Regulator", page 55.

### 3.2.118 FF\_OUT - Register 118

Data format: Integer.

This register contains output (velocity) from filter FF. See "Speed Regulator", page 55 and "KFF3 - Register 89", page 38.

### 3.2.119 VB\_OUT - Register 119

Data format: Integer.

This register contains output (velocity) from filter VB. See "Speed Regulator", page 55, "GEARB - Register 101", page 39, and "KVB3 - Register 102", page 39.

## 3.2 Parameter and Data Registers

---

### 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 57.
- G\_MODE. See "Gear Mode (G\_MODE, MODE = 3)", page 59.
- AVG\_MODE. See "Analog Velocity Gear mode (AVG\_MODE, MODE = 6)", page 61.

### 3.2.121 VF\_OUT - Register 121

Data format: Integer.

This register contains output (torque) from filter VF. See "Speed Regulator", page 55.

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 42.

The register is used in the following modes of operation:

- Analog Torque mode, AT\_MODE. See "Analog Torque Mode (AT\_MODE, MODE = 4)", page 60.
- Analog Velocity mode, AV\_MODE. See "Analog Velocity Mode", page 61.
- Analog Velocity Gear mode, AVG\_MODE. See "Analog Velocity Gear mode (AVG\_MODE, MODE = 6)", page 61.
- Home2 mode. See "HOME2\_MODE (MODE = 13)", page 65.
- Home3 mode. See "HOME3\_MODE (MODE = 14)", page 66.

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 42.

## 3.2 Parameter and Data Registers

---

### 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 43, "Current Regulator", page 56, and "CNTRL\_BITS - Register 36", page 31

### 3.2.126 PHASE\_COMP - Register 126

Data format: Fixed8.

Recommended range: 0.0 ... 1.0

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 43, "ELDEGP\_OFFSET - Register 125", page 43, "V\_ELDEG - Register 144", page 47 and "Current Regulator", page 56.

## 3.2 Parameter and Data Registers

---

### 3.2.127 AMPLITUDE - Register 127

Data format: Fixed4.

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:

$$1023 * \text{AMPLITUDE}$$

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 56 and "IA\_SOLL - Register 133", page 45.

### 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:

- $IA\_SOLL = MAN\_I\_NOM * \sin(MAN\_ALPHA) * AMPLITUDE$
- $IB\_SOLL = MAN\_I\_NOM * \sin(MAN\_ALPHA + 120 \text{ elect. degrees}) * AMPLITUDE$
- $IC\_SOLL = MAN\_I\_NOM * \sin(MAN\_ALPHA + 240 \text{ elect. degrees}) * AMPLITUDE$

Units for MAN\_I\_NOM: Dependent on motor type.

Units for MAN\_ALPHA:  $360 / 2048 \text{ ELECT. DEGREES} = 0.176 \text{ elect. degrees}$ .

See "Current Regulator", page 56, "IA\_SOLL - Register 133", page 45, and "AMPLITUDE - Register 127", page 44.

## 3.2 Parameter and Data Registers

---

### 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 62.

### 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 56 and "AMPLITUDE - Register 127", page 44.

### 3.2.136 IX\_SELECT - Register 136

Data format: Word.

Intended for debug purposes only.

## 3.2 Parameter and Data Registers

---

### 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 56.

### 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 57.

In INIT\_MODE, IA, IB and IC\_OFFSET are adjusted so that:

- $IA\_IST = I_{a\_measured} + IA\_OFFSET = 0.$
- $IB\_IST = I_{b\_measured} + IB\_OFFSET = 0.$
- $IC\_IST = I_{c\_measured} + IC\_OFFSET = 0.$

### 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 56, "*ELDEGN\_OFFSET - Register 124*", page 43, "*ELDEGP\_OFFSET - Register 125*", page 43, and "*CNTRL\_BITS - Register 36*", page 31.

## 3.2 Parameter and Data Registers

---

### 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 43 and "Current Regulator", page 56.

### 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 PWMA\_VAL - Register 148

### 3.2.149 PWMB\_VAL - Register 149

### 3.2.150 PWMC\_VAL - Register 150

Data format: Word.

Intended for debug purposes only.

### 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 56.

### 3.2.152 GFERR - Register 152

Data format: Word.

Intended for debug purposes only.

## 3.2 Parameter and Data Registers

---

### 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 72.



## 4 Program/Function Description

---

The program software contains:

- Function blocks for generation of velocity profiles. See "*Function blocks (Profile Generation)*", page 50.
- A velocity regulator. See "*Speed Regulator*", page 55.
- Current Regulator for the three phase currents. See "*Current Regulator*", page 56.
- Monitoring functions. See "*Monitoring Functions*", page 68.

"*Function blocks (Profile Generation)*", page 50 describes the various function blocks.

"*Speed Regulator*", page 55 explains the velocity/position regulator.

"*Current Regulator*", page 56 describes the current regulator.

The software controls the MAC-motor in several different modes of operation. "*Modes*", page 57 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 68.

## 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 50.
- Position function block. See "*Position function block*", page 50.
- Gear function block. See "*Gear function block*", page 52.
- Torque function block. See "*Torque function block*", page 53.
- AV-function block. See "*AV function block*", page 53.
- AVG-function block. See "*AVG function block*", page 54.
- Stop-function block. See "*Stop function block*", page 54.

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 41.
- ACC\_FLAG and DEC\_FLAG. See "*ERR\_STAT - Register 35*", page 30.
- FNC\_ERR. See "*FNCERR*", page 27.

The function can be described by:

- $DV = V\_SOLL - V\_OLD$
- Limit  $DV$  to  $\pm A\_SOLL$
- $FNC\_OUT = V\_OLD + DV$
- $ACC\_FLAG = FNC\_OUT > V\_OLD$
- $DEC\_FLAG = FNC\_OUT < V\_OLD$
- $V\_OLD = FNC\_OUT$

Registers DV and V\_OLD are not accessible to the user. V\_OLD is set to 0 in INIT\_MODE.

### 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 41.

## 4.1 Function blocks (Profile Generation)

---

- ACC\_FLAG and DEC\_FLAG. See "ERR\_STAT - Register 35", page 30.
- P\_FNC. See below. See "P\_FNC - Register 8", page 21.

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  $\pm V\_SOLL$ .
- $DV = V\_SOLL - V\_OLD$
- Limit DV to  $\pm A\_SOLL$ .
- $FNC\_OUT = V\_OLD + DV$
- $ACC\_FLAG = FNC\_OUT > V\_OLD$ .
- $DEC\_FLAG = FNC\_OUT < V\_OLD$
- $V\_OLD = FNC\_OUT$
- $P\_FNC = P\_FNC + FNC\_OUT$

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 21.

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 41

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.

See "P\_FNC - Register 8", page 21, and the RELPOSPSOLL and RELPOSPFNC bits under "ERR\_STAT - Register 35", page 30.

See "T\_SOLL - Register 7", page 20, "FNC\_OUT - Register 117", page 41, and "Speed Regulator", page 55.

## 4.1 Function blocks (Profile Generation)

---

### 4.1.3 Gear function block

This 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 42
- GEARF1, GEARF2: Gear factor =  $\text{GEARF1} / \text{GEARF2}$ .  
See "GEARF1 - Register 14", page 24.
- 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:

## 4.1 Function blocks (Profile Generation)

---

- Velocity profile is written to register: FNC\_OUT.  
See "FNC\_OUT - Register 117", page 41.
- 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:

- $FNC\_OUT = V\_EXT * GEARF1 / GEARF2$
- $P\_FNC = P\_FNC - FNC\_OUT$
- $T = (FNC\_OUT - V\_OLD) / A\_SOLL$ , (calculated deceleration time).
- $FNC\_OUT = -2 * P\_FNC / T$ .
- Limit FNC\_OUT to  $\pm V\_SOLL$ .
- $DV = V\_SOLL - V\_OLD$
- Limit DV to  $\pm A\_SOLL$ .
- $FNC\_OUT = V\_OLD + DV$
- $ACC\_FLAG = FNC\_OUT > V\_OLD$ .
- $DEC\_FLAG = FNC\_OUT < V\_OLD$
- $V\_OLD = FNC\_OUT$
- $P\_FNC = P\_FNC + FNC\_OUT$ .

See "FNC\_OUT - Register 117", page 41, "GEARF1 - Register 14", page 24, "GEARF2 - Register 15", page 24, "V\_SOLL - Register 5", page 19, "A\_SOLL - Register 6", page 20, and "CNTRL\_BITS - Register 36", page 31.

### 4.1.4 Torque function block

This function block generates a *torque* as follows:

- The analogue input is converted such that  $\pm 10V$  corresponds to  $\pm T\_SOLL$ .

The following registers are updated:

- Torque is written to register VF\_OUT. See "VF\_OUT - Register 121", page 42.

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.

### 4.1.5 AV function block

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

- The analogue input is converted such that  $\pm 10V$  corresponds to  $\pm V\_SOLL$ . See "ANINP - Register 122", page 42.
- A\_SOLL: Maximum acceleration / deceleration in order to achieve velocity.

## 4.1 Function blocks (Profile Generation)

---

The following registers are updated:

- Velocity profile is written to register: FNC\_OUT.  
See "FNC\_OUT - Register 117", page 41.

### 4.1.6 AVG function block

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

- GEARF1, GEARF2: Gear factor =  $\text{GEARF1} / \text{GEARF2}$ .  
See "GEARF1 - Register 14", page 24.
- 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 42.
- 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 41.

### 4.1.7 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 29.
- 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 58, "Position Mode", page 58, "Gear Mode (G\_MODE, MODE = 3)", page 59, "Analog Velocity Mode", page 61, and "Analog Velocity Gear mode (AVG\_MODE, MODE = 6)", page 61.

The following registers are updated:

- Velocity profile is written to register: FNC\_OUT.  
See "FNC\_OUT - Register 117", page 41.

## 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 41. 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 41, and "KFF3 - Register 89", page 38.

Encoder-counts are sampled and V\_IST are calculated as follows:

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

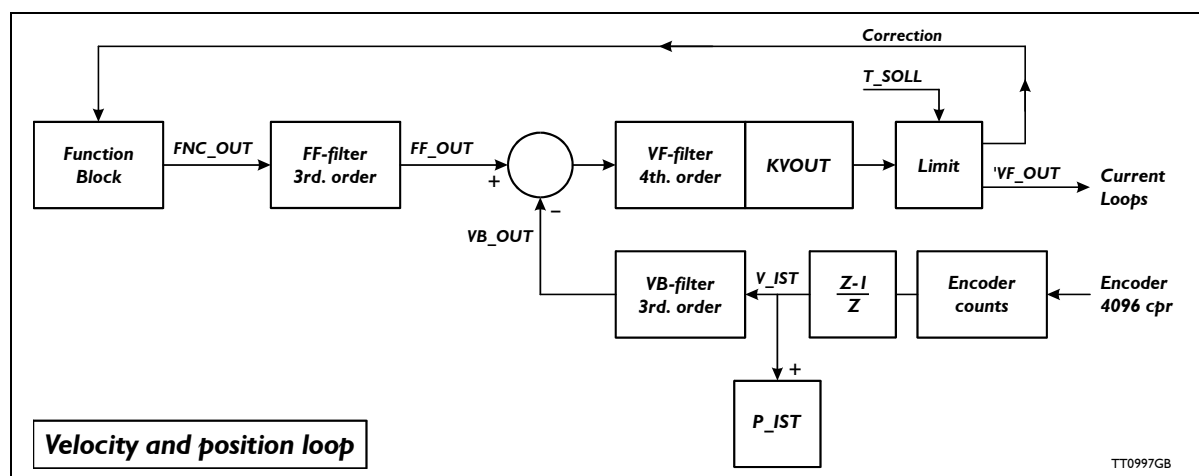
V\_IST is added to P\_IST.

V\_IST passes through the VB filter. See "VB\_OUT - Register 119", page 41. 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 38. The output of this filter is multiplied by KVOUT, and the result ( torque ) is written to register VF\_OUT. See "KVOUT - Register 13", page 23, and "VF\_OUT - Register 121", page 42.

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 21, and "FNCERR", page 27.

The calculated torque, VF\_OUT, is input for the current regulator. See "Current Regulator", page 56.



## 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 55.
- The torque function block. See "Torque function block", page 53.

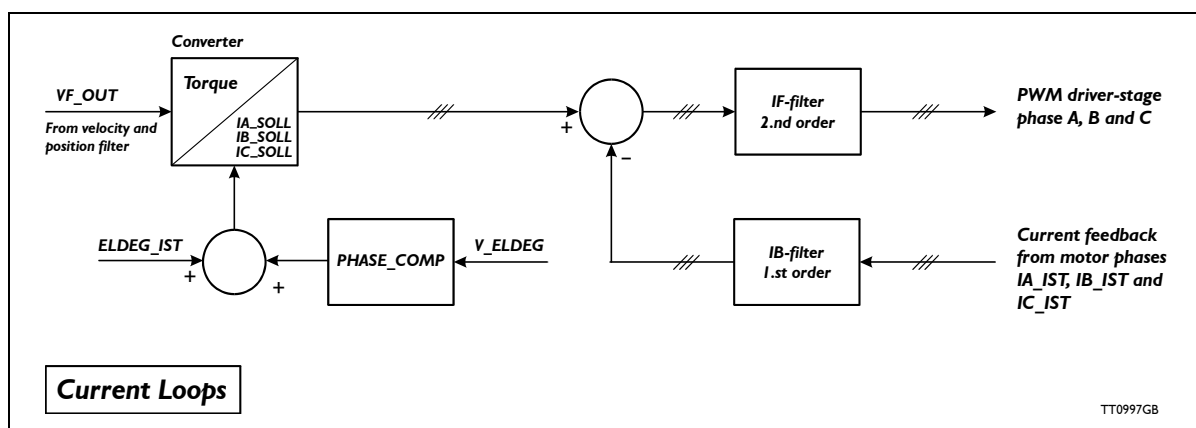
On the basis of VF\_OUT, ( torque ), the amplitude and commutation angle, ( PHI\_SOLL ), for the stator field are calculated. See "ELDEGN\_OFFSET - Register 124", page 43, "ELDEGP\_OFFSET - Register 125", page 43, "PHASE\_COMP - Register 126", page 43, "PHI\_SOLL - Register 132", page 45, "ELDEG\_IST - Register 143", page 46, and "V\_ELDEG - Register 144", page 47.

The three phase currents are calculated as follows:

- $IA\_SOLL = I\_NOM * AMPLITUDE * \sin( PHI\_SOLL )$
- $IB\_SOLL = I\_NOM * AMPLITUDE * \sin( PHI\_SOLL + 120 \text{ electr. degrees} )$
- $IC\_SOLL = I\_NOM * AMPLITUDE * \sin( PHI\_SOLL + 240 \text{ electr. degrees} )$

See "IA\_SOLL - Register 133", page 45 and "AMPLITUDE - Register 127", page 44.

The three phase currents are regulated as illustrated below. See also "KIFX2 - Register 106", page 39, and "KIB1 - Register 110", page 40, regarding the current filters.





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 mode, G\_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 )

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 25 and "*I2T*", page 68.
- UIT: Calculated load of internal power dump. See "*UIT - Register 18*", page 25 and "*UIT*", page 68.
- I2T\_ERR: Bit that indicates coil temperature overload. See "*ERR\_STAT - Register 35*", page 30 and "*I2T*", page 68.
- UIT\_ERR: Bit that indicates overload of internal power dump. See "*ERR\_STAT - Register 35*", page 30 and "*UIT*", page 68.
- P\_LIM\_ERR: Bit that indicates that a software position limit has been exceeded. See "*ERR\_STAT - Register 35*", page 30 and "*Software end-of-travel limits*", page 68.

#### 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 46.

The following registers and flags are updated:

- P\_IST. See "P\_IST - Register 10", page 21.
- V\_IST. See "V\_IST - Register 12", page 22.
- IN\_POS bit = 0. See "ERR\_STAT - Register 35", page 30, "INPOSWIN - Register 33", page 29, and "INPOSCNT - Register 34", page 29.
- FLWERR = 0. See "ERR\_STAT - Register 35", page 30, "FLWERR - Register 20", page 26, and "Follow error", page 69.
- FNCERR = 0. See "ERR\_STAT - Register 35", page 30, "FNCERR", page 27, and "Function Error", page 69.
- IA, IB and IC\_OFFSET are adjusted. See "IA\_OFFSET - Register 140", page 46.

#### 4.4.2 Velocity Mode

In Velocity Mode the configuration is as follows:

- Function blocks: Velocity function block. See "Velocity function block", page 50.
- 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 21.
- V\_IST. See "V\_IST - Register 12", page 22.
- FLWERR. Accumulated value of velocity error. See "ERR\_STAT - Register 35", page 30, "FLWERR - Register 20", page 26, and "Follow error", page 69.
- FNCERR. See "ERR\_STAT - Register 35", page 30, "FNCERR", page 27, and "Function Error", page 69.
- IN\_POS bit in ERRSTAT is set if  $|FLWERR| < INPOSWIN$ . See "ERR\_STAT - Register 35", page 30, "INPOSWIN - Register 33", page 29, "INPOSCNT - Register 34", page 29, and "Function blocks (Profile Generation)", page 50.
- FLW\_ERR. See "ERR\_STAT - Register 35", page 30.
- ACC\_FLAG. See "ERR\_STAT - Register 35", page 30 and "Velocity function block", page 50.
- DEC\_FLAG. See "ERR\_STAT - Register 35", page 30 and "Velocity function block", page 50.

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 50.
- Speed Regulator: Active. See "Speed Regulator", page 55.
- Current Regulator: Active. See "Current Regulator", page 56.

## 4.4

## Modes

- Correction of function block: Active. See "Position function block", page 50.

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 21.
- V\_IST. See "V\_IST - Register 12", page 22.
- FLWERR: Accumulated value of velocity error. See "ERR\_STAT - Register 35", page 30, "FLWERR - Register 20", page 26, and "Follow error", page 69.
- FNCERR: Accumulated value of correction of function block. See "ERR\_STAT - Register 35", page 30, "FNCERR", page 27, and "Function Error", page 69.
- IN\_POS bit in ERRSTAT is set if  $|P\_SOLL - P\_FNC/16 + FLWERR| < INPOSWIN$ . See "ERR\_STAT - Register 35", page 30, "INPOSWIN - Register 33", page 29, "INPOSCNT - Register 34", page 29, "P\_SOLL - Register 3", page 18, "P\_FNC - Register 8", page 21, "FLWERR - Register 20", page 26, and "Position function block", page 50.
- FNC\_ERR. See "ERR\_STAT - Register 35", page 30.
- FLW\_ERR. See "ERR\_STAT - Register 35", page 30.
- ACC\_FLAG. See "ERR\_STAT - Register 35", page 30 and "Position function block", page 50.
- DEC\_FLAG. See "ERR\_STAT - Register 35", page 30 and "Position function block", page 50.

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 Mode (G\_MODE, MODE = 3 )

In this mode of operation the configuration is as follows:

- Function block: Gear function block. See "Gear function block", page 52.
- Speed Regulator: Active. See "Speed Regulator", page 55.
- Current Regulator: Active. See "Current Regulator", page 56.
- Correction of function block: Active. See "Gear function block", page 52.

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 24.
- 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.

## 4.4

## Modes

- T\_SOLL: Maximum allowable torque to achieve required position. See "T\_SOLL - Register 7", page 20.

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

- P\_IST. See "P\_IST - Register 10", page 21.
- V\_IST. See "V\_IST - Register 12", page 22.
- FLWERR: Accumulated value of velocity error. See "ERR\_STAT - Register 35", page 30, "FLWERR - Register 20", page 26, and "Follow error", page 69.
- FNCERR: Accumulated value of correction of function block. See "ERR\_STAT - Register 35", page 30, "FNCERR", page 27, and "Function Error", page 69.
- IN\_POS bit in ERRSTAT is set if  $|P\_FNC| < INPOSWIN$ . See "ERR\_STAT - Register 35", page 30, "INPOSWIN - Register 33", page 29, "INPOSCNT - Register 34", page 29, and "Function blocks (Profile Generation)", page 50.
- FNC\_ERR. See "ERR\_STAT - Register 35", page 30.
- FLW\_ERR. See "ERR\_STAT - Register 35", page 30.

In this mode of operation, a change to INIT\_MODE always occurs via STOP\_MODE in order to protect the motor.

### 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 53.
- Speed Regulator: Passive.
- Current Regulator: Active. See "Current Regulator", page 56.
- Correction of function block: Passive

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

- T\_SOLL: Maximum torque, (  $\pm 10V$  is converted to  $\pm T\_SOLL$  ). See "ANINP - Register 122", page 42.

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

- P\_IST. See "P\_IST - Register 10", page 21.
- V\_IST. See "V\_IST - Register 12", page 22.
- FLWERR = 0. See "ERR\_STAT - Register 35", page 30.
- FNCERR = 0. See "ERR\_STAT - Register 35", page 30.
- IN\_POS = 0. See "ERR\_STAT - Register 35", page 30.

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

## Modes

---

### 4.4.6 Analog Velocity Mode

In this mode of operation the configuration is as follows:

- Function block: AV-function block. See "*AV function block*", page 53.
- Speed Regulator: Active. See "*Speed Regulator*", page 55.
- Current Regulator: Active. See "*Current Regulator*", page 56.
- 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, (  $\pm 10V$  analogue input is converted to  $\pm V\_SOLL$  ).
- 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 21.
- V\_IST. See "*V\_IST - Register 12*", page 22.
- FLWERR: Accumulated value of velocity error. See "*FLWERR - Register 20*", page 26.
- FNCERR = 0
- IN\_POS = 0. See "*ERR\_STAT - Register 35*", page 30.

In this mode of operation, a change to INIT\_MODE always occurs via STOP\_MODE in order to protect the motor.

### 4.4.7 Analog 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 54.
- Speed Regulator: Active. See "*Speed Regulator*", page 55.
- Current Regulator: Active. See "*Current Regulator*", page 56.
- 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 24.
- V\_SOLL: Maximum velocity for compensating for follow error.  
See "*V\_SOLL - Register 5*", page 19.
- ANINP:  $\pm 10V$  is converted to  $\pm 300$  RPM. See "*ANINP - Register 122*", page 42.
- 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 20.

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

- P\_IST. See "*P\_IST - Register 10*", page 21.
- V\_IST. See "*V\_IST - Register 12*", page 22.

## 4.4

## Modes

---

- FLWERR: Accumulated value of velocity error. See "FLWERR - Register 20", page 26.
- FNCERR = 0
- FLW\_ERR: Accumulated value of velocity error.  
See "ERR\_STAT - Register 35", page 30.
- IN\_POS = 0. See "ERR\_STAT - Register 35", page 30.

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 44.
- MAN\_ALPHA: Electrical commutation angle.  
See "MAN\_ALPHA - Register 129", page 44

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 "ELDEG\_OFFSET - Register 125", page 43, and "ELDEG\_IST - Register 143", page 46.

### 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 45.

This mode of operation performs the following function:

- SAMPLE1 = 145. ( Sample UA\_VAL to the sample buffer.  
See "SAMPLE1 - Register 112", page 41 )
- SAMPLE2 = 139. ( Sample IC\_IST to the sample buffer ).
- REWINDBIT = 1. ( Sample from start of buffer.  
See "CNTRL\_BITS - Register 36", page 31 )
- RECINNERBIT = 1. ( Sampling frequency = 7812 Hz.  
See "CNTRL\_BITS - Register 36", page 31 )
- RECORDBIT = 1. ( Start sampling. See "CNTRL\_BITS - Register 36", page 31 ).
- 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 41 )
- SAMPLE2 = 12. ( Sample V\_IST to the sample buffer ).
- REWINDBIT = 1. ( Sample from start of buffer.  
See "CNTRL\_BITS - Register 36", page 31 )
- RECINNERBIT = 0. ( Sampling frequency = 520.8 Hz.  
See "CNTRL\_BITS - Register 36", page 31 )
- RECORDBIT = 1. ( Start sampling. See "CNTRL\_BITS - Register 36", page 31 ).
- Sampling is carried out for 11 sampling periods.
- Switch to BREAK\_MODE. See "BREAK\_MODE ( MODE = 10 )", page 64.

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

## 4.4

## Modes

---

### 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 63.

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 54.
- Speed Regulator: Active. See "*Speed Regulator*", page 55.
- Current Regulator: Active. See "*Current Regulator*", page 56.
- 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 29.

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 57. A switch of operating mode to INIT\_MODE may occur as a result of a command or as a result 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 30 and "ACC\_EMERG - Register 32", page 29.

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 50.
- Speed Regulator: Active. See "*Speed Regulator*", page 55.
- Current Regulator: Active. See "*Current Regulator*", page 56.
- Correction of function block: Passive



Parameters:

- V\_HOME: Velocity used in conjunction with zero-point search.  
See "V\_HOME - Register 40", page 32.
- T\_HOME: Torque limit for detecting mechanical limit.  
See "T\_HOME - Register 41", page 33
- P\_HOME: Defined positional value for zero-point.  
See "P\_HOME - Register 38", page 32.
- 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 21.
- V\_IST. See "V\_IST - Register 12", page 22.
- FLWERR: Accumulated value of velocity error. See "FLWERR - Register 20", page 26.
- FNCERR = 0
- FLW\_ERR. See "ERR\_STAT - Register 35", page 30.
- IN\_POS = 0. See "ERR\_STAT - Register 35", page 30.

This mode of operation performs the following function:

- V\_SOLL = V\_HOME. See "V\_HOME - Register 40", page 32.
- T\_SOLL = 1.5 \* T\_HOME. (Allow higher peak torque than detection limit). See "T\_HOME - Register 41", page 33.
- 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 42.
- 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 21.
- P\_FNC = P\_HOME \* 16. See "P\_FNC - Register 8", page 21.
- P\_SOLL = 0
- T\_SOLL = original value.
- MODE\_REG = STARTMODE. See "MODE\_REG - Register 2", page 17, and "START-MODE - Register 37", page 32.

#### 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 50.
- Speed Regulator: Active. See "Speed Regulator", page 55.
- Current Regulator: Active. See "Current Regulator", page 56.
- Correction of function block: Passive

Parameters:

- V\_HOME: Velocity used in conjunction with zero-point search.  
See "V\_HOME - Register 40", page 32.
- T\_SOLL: Maximum torque used during search. See "T\_SOLL - Register 7", page 20.
- P\_HOME: Defined positional value for zero point. See "P\_HOME - Register 38", page 32.

## 4.4

## Modes

- 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 33.

The following registers and flags are updated in HOME2\_MODE:

- P\_IST. See "P\_IST - Register 10", page 21.
- V\_IST. See "V\_IST - Register 12", page 22.
- FLWERR: Accumulated value of velocity error. See "FLWERR - Register 20", page 26.
- FNCERR = 0
- FLW\_ERR. See "ERR\_STAT - Register 35", page 30.
- IN\_POS = 0. See "ERR\_STAT - Register 35", page 30.

This mode of operation performs the following function:

- V\_SOLL = -V\_HOME, ( Move away from home sensor ).  
See "V\_HOME - Register 40", page 32.
- 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 21.
- P\_FNC = P\_HOME \* 16. See "P\_FNC - Register 8", page 21.
- P\_SOLL = 0
- MODE\_REG = STARTMODE. See "MODE\_REG - Register 2", page 17,  
and "STARTMODE - Register 37", page 32.

### 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 50.
- Speed Regulator: Active. See "Speed Regulator", page 55.
- Current Regulator: Active. See "Current Regulator", page 56.
- Correction of function block: Passive

Parameters:

- V\_HOME: Velocity used in conjunction with zero-point search. See "V\_HOME - Register 40", page 32.
- T\_SOLL: Maximum torque used during search. See "T\_SOLL - Register 7", page 20.
- P\_HOME: Defined positional value for zero position. See "P\_HOME - Register 38", page 32.
- 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 33.

The following registers and flags are updated in HOME3\_MODE:

- P\_IST. See "P\_IST - Register 10", page 21.
- V\_IST. See "V\_IST - Register 12", page 22.
- FLWERR: Accumulated value of velocity error. See "FLWERR - Register 20", page 26.
- FNCERR = 0

- FLW\_ERR. See "ERR\_STAT - Register 35", page 30.
- IN\_POS = 0. See "ERR\_STAT - Register 35", page 30.

This mode of operation performs the following function:

- $V\_SOLL = -V\_HOME$ , ( Move away from home sensor ). See "V\_HOME - Register 40", page 32.
- 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 21.
- $P\_FNC = P\_HOME * 16$ . See "P\_FNC - Register 8", page 21.
- $P\_SOLL = 0$
- $MODE\_REG = STARTMODE$ . See "MODE\_REG - Register 2", page 17, and "STARTMODE - Register 37", page 32.

#### 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.

## 4.5

## Monitoring Functions

---

The motor software contains 5 monitoring functions, which are always active:

- Monitoring of the motor's coil temperature ( calculated ). See "*I<sup>2</sup>T*", page 68.
- Monitoring of the internal "power dump" temperature, ( calculated ). See "*UIT*", page 68.
- Monitoring of the software end-of-travel limits. See "*Software end-of-travel limits*", page 68.
- Monitoring of the "Follow error". See "*Follow error*", page 69.
- Monitoring of Function error. See "*Function Error*", page 69.

### 4.5.1 I<sup>2</sup>T

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 25, "*I2TLIM - Register 17*", page 25, and "*ERR\_STAT - Register 35*", page 30.

**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 25, "*UITLIM - Register 19*", page 25, and "*STOP\_MODE ( MODE = 11 )*", page 64.

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 21.

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  $\neq$  0 or register MAX\_P\_IST  $\neq$  0. See "*P\_IST - Register 10*", page 21, and "*MIN\_P\_IST - Register 28*", page 28.

When active, the monitoring function will set PLIM\_ERR ( 1 ) if:

## 4.5 Monitoring Functions

---

- $P\_IST < MIN\_P\_IST$ .
- $P\_IST > MAX\_P\_IST$ .

See "ERR\_STAT - Register 35", page 30. 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 64.

### 4.5.4 Follow error

The FLWERR register is updated in several modes of operation. See "FLWERR - Register 20", page 26. 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 26.

When active, the monitoring function will set FLW\_ERR ( 1 ) if:

- Absolute value ( FLW\_ERR ) > FLWERRMAX.

See "ERR\_STAT - Register 35", page 30. 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 64.

### 4.5.5 Function Error

The FNCERR register is updated in several modes of operation. See "FNCERR", page 27. 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 27.

When active, the monitoring function will set FNC\_ERR ( 1 ) if:

- Absolute value ( FNC\_ERR ) > FNCERRMAX.

See "ERR\_STAT - Register 35", page 30. 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 64.





## 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 48 )  
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
```



## 5.1

## MacTalk Protocol

---

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 31.

The FastMAC-protocol is designed to achieve rapid communication by limiting all commands and responses to 1 byte. On the other 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 34 to "Z4 - Register 88", page 37.

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 registers for controlling the motor in register mode:

- 8 position registers: P[ 1 .. 8 ]. See "P1 - Register 49", page 36, and "P\_REG\_P - Register 43", page 34.
- 8 velocity registers: V[ 1 .. 8 ]. See "V1 - Register 65", page 36, and "V\_REG\_P - Register 44", page 34.
- 4 acceleration registers: A[ 1 .. 4 ]. See "A1 - Register 73", page 37 and "A\_REG\_P - Register 45", page 34.
- 4 torque registers: T[ 1 .. 4 ]. See "T1 - Register 77", page 37, and "T\_REG\_P - Register 46", page 35.
- 4 load registers: L[ 1 .. 4 ]. See "L1 - Register 81", page 37, and "L\_REG\_P - Register 47", page 35.
- 4 "In\_position\_registers": Z[ 1 .. 4 ]. See "Z1 - Register 85", page 37, and "Z\_REG\_P - Register 48", page 35.

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 57.
- **Velocity-mode:** In this mode the register set is interpreted as follows: ( See "Velocity Mode", page 58 )  
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 58 )  
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.

## 5.2

## FastMAC / FlexMAC

---

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 31.

### 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 31, 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 30, and "Position Mode", page 58.

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 30, and "Position Mode", page 58.
- 03H: P\_IST = 0.
- 04H: P\_FNC = 0. ( step turntable ). IN\_POS = 0. See "P\_FNC - Register 8", page 21, "Position function block", page 50, "ERR\_STAT - Register 35", page 30, and "Position Mode", page 58.
- 05H: V\_SOLL = 0. See "V\_SOLL - Register 5", page 19.
- 06H: T\_SOLL = 0. See "T\_SOLL - Register 7", page 20.
- 07H: IN\_POS-flag = 0. See "ERR\_STAT - Register 35", page 30, "Velocity function block", page 50, "Position function block", page 50, and "Gear function block", page 52.
- 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 50, "P\_FNC - Register 8", page 21 and "FLWERR - Register 20", page 26
- 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 50, "P\_FNC - Register 8", page 21 and "FLWERR - Register 20", page 26
- 0AH: Use rapid communication ( based on 1 byte ). See "CNTRL\_BITS - Register 36", page 31.
- 0BH: Use safe communication ( based on 2 identical bytes ). See "CNTRL\_BITS - Register 36", page 31.
- 0CH: Use register set: P1, V1, A1, T1, L1, Z1. See sections "P\_REG\_P - Register 43", page 34 to "Z\_REG\_P - Register 48", page 35. IN\_POS = 0. See "ERR\_STAT - Register 35", page 30, and "Position Mode", page 58.
- 0DH: Use register set: P2, V2, A2, T2, L2, Z2. See sections "P\_REG\_P - Register 43", page 34 to "Z\_REG\_P - Register 48", page 35. IN\_POS = 0. See "ERR\_STAT - Register 35", page 30, and "Position Mode", page 58.
- 0EH: Use register set: P3, V3, A3, T3, L3, Z3. See sections "P\_REG\_P - Register 43", page 34 to "Z\_REG\_P - Register 48", page 35. IN\_POS = 0. See "ERR\_STAT - Register 35", page 30, and "Position Mode", page 58.
- 0FH: Use register set: P4, V4, A4, T4, L4, Z4. See sections "P\_REG\_P - Register 43", page 34 to "Z\_REG\_P - Register 48", page 35. IN\_POS = 0. See "ERR\_STAT - Register 35", page 30, and "Position Mode", page 58.
- 10H: Home search. See "HOME\_MODE - Register 42", page 33, and sections "HOME1\_MODE ( MODE = 12 )", page 64 to "HOME3\_MODE ( MODE = 14 )", page 66.
- 11H: NOP.
- 12H: NOP.
- 13H: Transmit slowly. See "Synchronisation", page 79.
- 14H: Use absolute positioning using position registers.  
i.e. Bit RELPOSPSOLL = 0, bit RELPOSPFNC = 0. See "Position function block", page 50, "ERR\_STAT - Register 35", page 30 and "P\_REG\_P - Register 43", page 34.
- 15H: Use relative positioning using position registers.  
i.e. Bit RELPOSPSOLL = 1, bit RELPOSPFNC = 0. See "Position function block", page 50, "ERR\_STAT - Register 35", page 30 and "P\_REG\_P - Register 43", page 34.
- 16H: Use relative positioning using position registers.  
i.e. Bit RELPOSPSOLL = 0, bit RELPOSPFNC = 1. See "Position function block", page 50, "ERR\_STAT - Register 35", page 30 and "P\_REG\_P - Register 43", page 34.

## 5.2

## FastMAC / FlexMAC

---

- I7H: NOP
- I8H: NOP
- I9H: NOP
- IAH: NOP
- IBH: NOP
- ICH: NOP
- IDH: NOP
- IEH: Use FASTMAC protocol, i.e. NOP
- IFH: Use FLEXMAC protocol

## 5.2

## FastMAC / FlexMAC

---

### 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 31.
- Bit 6: Deceleration flag. See "*ERR\_STAT - Register 35*", page 30, "*Velocity function block*", page 50, and "*Position function block*", page 50.
- Bit 5: Acceleration flag. See "*ERR\_STAT - Register 35*", page 30, "*Velocity function block*", page 50, and "*Position function block*", page 50.
- Bit 4: In\_Position\_flag: This flag is updated in Position\_mode. See "*INPOSWIN - Register 33*", page 29, "*INPOSCNT - Register 34*", page 29, and "*ERR\_STAT - Register 35*", page 30.
- Bit 3: Active protocol, ( receive ): FastMAC = 0, FlexMAC = 1. See "*CNTRL\_BITS - Register 36*", page 31, and "*FlexMAC*", page 79.
- Bit 2: Character received with framing error. See "*Synchronisation*", page 79.
- Bit 1: Available.
- Bit 0: Follow / Function / I2T / UIT – error: Read ERR\_STAT using FlexMAC. See "*ERR\_STAT - Register 35*", page 30.

**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 75, 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 30. In addition, the MAC motor will transmit status bytes with bit 2 set ( character received with framing error ). See "Status", page 78. 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 >

## 5.2

## FastMAC / FlexMAC

---

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: ERBIT( word )
- 14: Command: Use FastMAC( word )
- 15: Command: Use FlexMAC( word ) ( NOP )

## 5.2

## FastMAC / FlexMAC

---

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 in status byte = 0
- Transmit using FastMAC.



**A**

A\_REG\_P 34, 74  
 A\_SOLL 20, 50, 52–54, 58–59, 61, 64–66  
 A1 37  
 A2 37  
 A3 37  
 A4 37  
 ACC\_EMERG 29  
 ACC\_FLAG 50–51, 58–59, 83  
 AMPLITUDE 44, 56  
 Analog mode 57, 60  
 Analog velocity mode 57, 61  
 ANINP 42, 53–54, 61  
 ANINP\_OFFSET 42  
 AV function block 53  
 AVG function block 54

**B**

Bit window 11  
 BREAK\_MODE mode 57, 63–64

**C**

Clear Samples 12  
 CNTRL\_BITS 31  
 Current regulator 56

**D**

Data formats 16  
 DEC\_FLAG 50–51, 58–59, 83  
 DP 51  
 DV 50–51

**E**

ELDEG\_IST 46  
 ELDEGN\_OFFSET 43, 56  
 ELDEGP\_OFFSET 43, 56  
 ERR\_STAT 30, 50–51

**F**

FastMAC 74  
 FastMAC/FlexMAC protocol 74–76, 78–83  
 FF\_OUT 41, 55  
 FF-filter 55  
 Fixed4 16  
 Fixed8 16  
 FLW\_ERR 58–60, 62, 65–67, 69  
 FLWERR 26, 58–62, 65–66, 69  
 FLWERRMAX 26, 69  
 FNC\_ERR 50, 59–60, 69  
 FNC\_OUT 41, 50, 53–55  
 FNCERR 27, 58–62, 65–66, 69  
 FNCERRMAX 27, 69  
 Follow error 69  
 Function blocks 50–51, 53–54  
     AV function block 53

AVG function block 54  
 Gear function block 52  
 Position function block 50  
 Stop function block 54  
 Torque function block 53  
 Velocity function block 50

Function error 69

**G**

Gear factor 52, 54, 59, 61  
 Gear function block 52  
 Gear mode 57, 59  
 Gear mode + Analog velocity mode 57, 61  
 GEARB 39  
 GEARF1 24, 52, 54, 59, 61  
 GEARF2 24, 52, 54, 59, 61  
 GFERR 47

**H**

HOME\_MODE 33  
 Home1 mode 57, 64  
 Home2 mode 57, 65  
 Home3 mode 57, 66

**I**

I\_NOM 45  
 I2T 25, 57, 68  
 I2T\_ERR 57, 68  
 I2TLIM 25  
 I2TMAX 68  
 IA\_IST 46  
 IA\_OFFSET 46, 58  
 IA\_SOLL 45, 56  
 IB\_IST 46  
 IB\_OFFSET 46, 58  
 IB\_SOLL 45, 56  
 IC\_IST 46  
 IC\_OFFSET 46, 58  
 IC\_SOLL 45, 56  
 IN\_POS 58–62, 65–67  
 Init mode 57  
 INIT\_MODE 54  
 INPOSCNT 29  
 INPOSWIN 29  
 Integer 16  
 Interface description 71–76, 78–83  
 IX\_SELECT 45

**K**

KFF0 38  
 KFF1 38  
 KFF2 38  
 KFF3 38, 55  
 KIB0 40  
 KIB1 40, 56

- 
- KIFX1 39
  - KIFX2 39, 56
  - KIFY0 39
  - KIFY1 39
  - KVB0 39
  - KVB1 39
  - KVB2 39
  - KVB3 39
  - KVFX1 38
  - KVFX2 38
  - KVFX3 38
  - KVFX4 38, 55
  - KVFX0 38
  - KVFX1 38
  - KVFX2 38
  - KVFX3 38
  - KVOUT 23, 55
  - L**
  - L\_REG\_P 35, 74
  - L1 37
  - L2 37
  - L3 37
  - L4 37
  - LongInt 16
  - M**
  - MacRegIO 9–11, 13
    - Bit window 11
    - Clear samples button 12
    - Enter Safe Mode 12
    - Exit safe mode 12
    - Installation 10
    - Operation 13
    - Parameter window 11
    - Receive window 11
    - Sample Window 11
    - Screen windows 11
    - Software reset button 12
    - Transmit window 11
    - Write to flash button 12
  - MacTalk Protocol 72
  - MAN\_ALPHA 44, 62
  - MAN\_I\_NOM 44, 62
  - Manual mode 57, 62
  - MAX\_P\_IST 28, 68
  - MIN\_P\_IST 28, 68
  - MODE\_REG 17
  - Modes 17, 57–67
    - Analog mode 57, 60
    - Analog velocity mode 57, 61
    - BREAK\_MODE mode 57, 64
    - Gear mode 57, 59
    - Gear mode + Analog velocity mode 57, 61
    - Home1 mode 57, 64
    - Home2 mode 57, 65
    - Home3 mode 57, 66
    - Init mode 57
    - Manual mode 57, 62
    - Position mode 57–58
    - Programming mode 57, 67
    - Safe mode 67
    - STOP\_MODE mode 57, 64
    - TESTA\_MODE mode 57, 63
    - TESTU\_MODE mode 57, 62
    - Velocity mode 57–58
  - Monitoring functions 68–69
    - Follow error 69
    - Function error 69
    - I2T 68
    - Software end-of-travel limits 68
    - UIT 68
  - MOTOR\_TYPE 48
  - MYADDR 48
  - O**
  - Operation modes 17, 57–67
    - Analog mode 57, 60
    - Analog velocity mode 57, 61
    - BREAK\_MODE mode 57, 64
    - Gear mode 57, 59
    - Gear mode + Analog velocity mode 57, 61
    - Home1 mode 57, 64
    - Home2 mode 57, 65
    - Home3 mode 57, 66
    - Init mode 57
    - Manual mode 57, 62
    - Position mode 57–58
    - Programming mode 57, 67
    - Safe mode 67
    - STOP\_MODE mode 57, 64
    - TESTA\_MODE mode 57, 63
    - TESTU\_MODE mode 57, 62
    - Velocity mode 57–58
  - P**
  - P\_FNC 21, 51, 53
  - P\_HOME 32, 65–66
  - P\_IST 21, 55, 58–62, 64–66
  - P\_LIM\_ERR 57
  - P\_REG\_P 34, 74
  - P\_SOLL 50, 59
  - PI 36
  - P2 36
-

- 
- P3 36
  - P4 36
  - P5 36
  - P6 36
  - P8 36
  - Parameter and Data Description 15
  - Parameter and Data Registers 17–48
    - Current loop registers 43–47
    - Data acquisition registers 41
    - Diverse registers 48
    - Error handling registers 25–30
    - Filter registers 38–40
    - Main control registers 17–24
    - Position/velocity loop registers 41–42
    - Power on registers 31–33
    - Register mode registers 34–37
  - Parameter window 11
  - PHASE\_COMP 43, 56
  - PHI\_SOLL 45, 56
  - Position function block 50
  - Position mode 57–58
  - PROG\_VERSION 17
  - Program and Function Description 49
  - Programming mode 57, 67
  - PWMA\_VAL 47
  - PWMB\_VAL 47
  - PWMC\_VAL 47
  - R**
  - REC\_CNT 41
  - Receive window 11
  - RECINNERBIT 63
  - RECORDBIT 63
  - Register mode 74
  - REWINDBIT 63
  - S**
  - Safe Mode 12
  - Safe mode 67
  - Sample Window 11
  - SAMPLE1 41, 63
  - SAMPLE2 41, 63
  - SAMPLE3 41
  - SAMPLE4 41
  - Sampled systems 7
  - Sampling frequency 7
  - Sampling interval 7
  - Sampling time 7
  - SERIAL\_NMB 48
  - Software end-of-travel limits 68
  - Software reset 12
  - Speed regulator 55
  - STARTMODE 32
  - Status 78
  - Stop function block 54
  - STOP\_MODE mode 57, 64
  - Synchronisation 79
  - T**
  - T 51
  - T\_HOME 33, 65–66
  - T\_REG\_P 35, 74
  - T\_SOLL 20, 55, 58–61, 63–66
  - T1 37
  - T2 37
  - T3 37
  - T4 37
  - TESTA\_MODE mode 57, 63
  - TESTU\_MODE mode 57, 62
  - Toggle bit 75
  - Torque function block 53
  - Transmit window 11
  - U**
  - U\_SUPPLY 47
  - UA\_VAL 47, 63
  - UB\_VAL 47
  - UC\_VAL 47, 63
  - UIT 25, 57, 68
  - UIT\_ERR 57, 68
  - UITLIM 25, 68
  - UITMAX 68
  - UMEAS 45, 63
  - V**
  - V\_ELDEG 47
  - V\_EXT 42, 55
  - V\_HOME 32, 65–66
  - V\_IST 22, 55, 58–62, 64–66
  - V\_OLD 50–51
  - V\_REG\_P 34, 74
  - V\_SOLL 19, 50, 52, 54, 58–59, 61, 65
  - V1 36
  - V2 36
  - V3 36
  - V4 36
  - V5 36
  - V6 36
  - V7 36
  - V8 36
  - VB filter 55
  - VB\_OUT 41, 55
  - Velocity function block 50
  - Velocity mode 57–58
  - VF filter 55
  - VF\_OUT 42, 53, 55–56
-

**W**

Word 16

Write to Flash 12

**Z**

Z\_REG\_P 35, 74

Z1 37

Z2 37

Z3 37

Z4 37