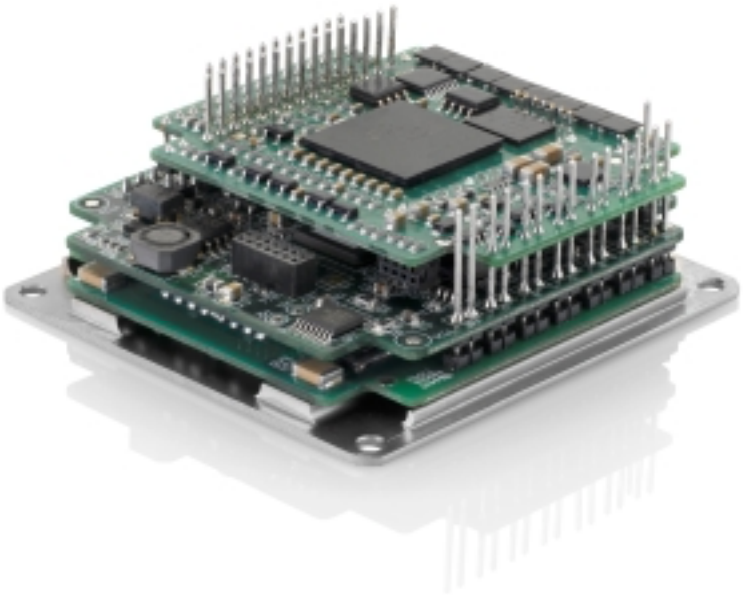
This PDF contains two manuals:

1. The SimplIQ for Steppers Command Reference Manual.
2. The SimplIQ for Steppers Application Note.

There are hyperlinks from the Command Reference to the Application Note.

SimplIQ for Steppers

Command Reference Manual



Ver. 1.1 - June 2009

Elmo
Motion Control

DFT

Important Notice

This guide is delivered subject to the following conditions and restrictions:

- This guide contains proprietary information belonging to Elmo Motion Control Ltd. Such information is supplied solely for the purpose of assisting users of *SimpIIQ for Steppers* servo drives.
- The text and graphics included in this manual are for the purpose of illustration and reference only. The specifications on which they are based are subject to change without notice.
- Information in this document is subject to change without notice.

Revision History

Ver. 1.1	June 2009	Updates
Ver. 1.0	Feb 2008	First revision.

Elmo Motion Control Ltd.	**Elmo Motion Control Inc.**	**Elmo Motion Control GmbH**	
64 Gisin St., P.O. Box 463 Petach Tikva 49103 Israel	42 Technology Way Nashua, NH 03060 USA	Steinkirchring 1 D-78056, Villingen-Schwenningen Germany	
Tel: +972 (3) 929-2300 Fax: +972 (3) 929-2322	Tel: +1 (603) 821-9979 Fax: +1 (603) 821-9943	Tel: +49 (0) 7720-85 77 60 Fax: +49 (0) 7720-85 77 70	
info-il@elmomc.com	info-us@elmomc.com	info-de@elmomc.com	**www.elmomc.com**

Contents

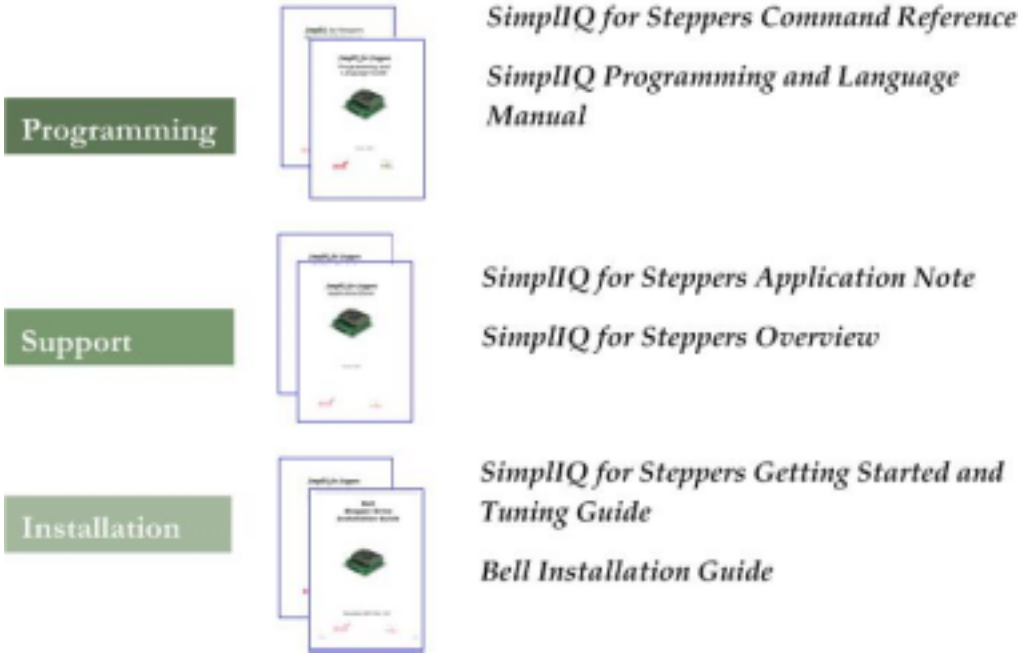
Chapter 1: Introduction

This manual describes, in detail, each software command used to manipulate the *SimpIIQ for Steppers* line of digital servo drives. It is an integral part of the *SimpIIQ for Steppers* documentation set, which includes:

- The Bell Stepper Drive Installation Guide, which provides full instructions for installing a drive.
- The Composer *User Manual*, which includes explanations of all the software tools that are a part of Elmo's Composer software environment.

The *SimpIIQ for Steppers* has been derived from Elmo's *SimpIIQ* interface. Most of the commands and explanations are identical. There are, however, some changes, following the introduction of stepper capabilities. We have tried to keep the changes minimal.

The following diagram illustrates the available *SimpIIQ for Steppers* documentation.



Note: This command reference does not describe commands that are only intended for service vendor-supplied tuning and setup tools.

Commands that are normally used by service vendor-supplied tuning and setup tools, but in some situations may be used by advanced users, are marked with this "Wizard's wand" symbol.

1.1. Command Specification

Commands for *SimpIIQ for Steppers* drives may be specified from the following sources:

- User program — A program loaded to the servo drive via one of the communication options. After program execution begins, the program is managed by the drive.
- RS-232 — Serial, point-to-point, short-range communication. Although this method is rather slow, RS-232 is very easy to use and the requirements are minimal: a standard PC with a serial port and ASCII terminal software.
- CANopen — Serial, multi-drop, medium speed and medium-range communication. This type of communication requires special-purpose host hardware and software.

This manual describes the *SimpIIQ for Steppers* commands that can be specified from each of these sources. Most of the commands are equally available for all three sources. Certain commands, however, are limited in scope according to the type of program or communication.

All the commands are available to CAN communication in text form through the OS service objects 0x1023 and 0x1024. In addition, the numerical set/get commands are available to CAN users in short PDO form, called the "binary interpreter." The binary and the OS SCAN interpreters are described fully in the *CANopen Implementation Guide*.

CANopen may also be used to manipulate the drive using the object dictionary (OD) method, which is the native CAN method. This manual does not cover OD manipulations with CANopen; refer to the "Object Dictionary" section of the *CANopen Implementation Guide* for full explanations.

The *SimpIIQ for Steppers* drive responds to many privileged commands — such as those used by the Composer setup wizard — that are not documented in this manual.

1.2. Scope

This manual includes the complete list of commands used by *SimpIIQ for Steppers* servo drives.

It specifies how to use each command, along with added remarks and examples.

The first part of this manual describes the communicated commands and their response.

The second part describes the user program and its control logic.

The commands are presented in two ways:

- A task-related listing
- Alphabetically

In the task-related reference, the commands are sorted into groups of related commands. Each group is presented in a table listing the commands with basic descriptions. The alphabetical command listing provides a detailed explanation of each command, with examples and references to the *SimplIQ for Steppers Application Note* when necessary.

This *Command Reference Manual* covers in addition the following topics:

- User program keywords, used for writing user programs. These, as well as other issues of developing, running and debugging user programs, are covered in the *SimplIQ for Steppers Application Note.*
- Interpreter functions and operators. The *SimplIQ for Steppers* interpreter allows complex arithmetic expressions and supports many arithmetic, trigonometric and logical operators. The syntax for interpreter commands is explained in the "Interpreter Language" chapter of the *SimplIQ for Steppers Application Note.*

1.3. Reserved Commands

SimplIQ for Steppers recognizes some reserved commands. The reserved commands serve the setup and tuning support software, or are intended for future implementation.

Some mnemonics are no longer used but they remain so that users can run legacy applications written for the SimplIQ series.

The reserved commands appear in the following table:

Command mnemonic	**Used for**
AB[N]	Absolute encoder setting parameters
AR[N]	Active recording session - Special Wizard commands
BH	Get a sample signal as hexadecimal
CC	Compile program
DF	Download firmware
DL	Receive a program downloaded from the host computer to Metronome. Can be used only in Composer software.
HP	Halt program execution
LP[N]	List parameters
LS	List program
PK	Peek memory
RC	Variables to record (two variables at each recording sequence)
RG	Recording gap, in samples. Gap between consecutive data recordings.
RL	Record length
RP[N]	Recorder parameters
RR	Recording on/off
RV[N]	Recorded variables

Command mnemonic	**Used for**
SI	Synchronize correction tables
TS	Obsolete – see TT[N]
TV[N]	Correction tables data
TW[N]	Wizard command
VC[N]	Phase voltage command - Special wizard commands
WI[N]	Metronome data, mainly for use by the Composer.
XA[N]	Obsolete – see PF[N]
XC	Continue program execution from current pointer, optionally until a given breakpoint
XP[N]	Obsolete – see PF[N], PN[N])
ZP[N]	Frequency response identification
ZX[N]	User program and auto-tuning temporary storage

Chapter 2: Functional Listing

This chapter summarizes the *SimpIIQ for Steppers* commands according to the following functional groups:

- Motion — Motion parameters, type and status. Begin/stop motion.
- I/O — Set outputs and report inputs.
- Status — Report Metronome status.
- Feedback — Support the multi-featured feedback interfaces.
- Configuration — Servo drive and motor types, and limitations.
- Communication — Communication type and parameters.
- Control filters — Digital, torque, speed and position filters.
- Protections — Failure and protection definitions.
- Data recording — Recording of internal Metronome variables for analysis.
- User programs — Application programming
- General — Miscellaneous commands.

Commands associated with more than one group are listed more than once.

2.1 Motion Commands

Command	Description	Page
AC	Acceleration, in counts per second^2	14
BG	Begin motion	21
BT	Begin motion at defined time	24
DC	Deceleration, in counts per second^2	38
HT[N]	Stepper mode holding torque	70
IL[N]	Input logic, defining how dedicated inputs behave	79
JV	Speed of jogging motion, in counts per second^2	88
MO	Motor on/off	106
PA	Absolute position reference for point-to-point motion	120
PR	Relative position reference for point-to-point motion	134
SD	Stop deceleration	152
SF	Smooth factor for motion command	154
SP	Speed for point-to-point motion	156
ST	Stop motion using deceleration value	159
TC	Torque command	161

2.2 I/O Commands

Command	Description	Page
AN[N]	Read analog inputs	18
IB[N]	Bit-wise digital input	76
IF[N]	Digital input filter	78
IP	Read all digital inputs	86
OB[N]	Bit-wise digital output	112
OC[N]	Output Compare	114
OL[N]	Output Logic	117
OP	Set all digital outputs	119

2.3 Status Commands

Command	Description	Page
BV	Maximum motor DC voltage	25
DD	CAN controller status	39
DV[N]	Reference desired value	42
EC	Error code: get code for last interpreter error	44
LC	Current limitation: report status of current limitation algorithm	94
MF	Motor fault: code for last motor-disable cause	101
MS	Motion status reporting	111
PK	Peak memory	123
SN	Serial number	155
SR	Numerical, bit-coded Metronome status	157
TI[N]	Temperature indications array	162
VR	Software (firmware) version	175

2.4 Feedback Commands

Command	Description	Page
AB[N]	Absolute encoder setting parameters	14
ID	Read active current	77

Command	Description	Page
IQ	Read reactive current	77
PE	Position error	122
PX	Main encoder position, in counts	139
PY	Auxiliary position	140
VE	Velocity error, in counts per second^2	173
VX	Main encoder velocity, in counts per second^2	176
VY	Velocity of auxiliary feedback	176
YA[N]	Auxiliary position sensor parameters	186

2.5 Configuration Commands

Command	Description	Page
AG[N]	Analog gains array	16
AS[N]	Analog input offsets array	20
BP[N]	Brake parameter	23
CA[N]	Commutation parameters array	26
CL[N]	Current continuous limitations array	35
EF[N]	Encoder filter frequency	57
EM[N]	ECAM parameters	58
ET[N]	Entries for ECAM table	62
FF[N]	Feed forward	63
FR[N]	Follower ratio	64
HM[N]	Homing and capture mode	67
HY[N]	Auxiliary home and capture mode	73
MC	Define maximum peak current of servo drive, in Amperes	99
MP[N]	Motion (PT/PVT) parameters	109
PL[N]	Peak duration and limit	127
PM	Profiler mode	129
PT	Position time command	136
PV	Position velocity time command	137
PW[N]	PWM signal parameters	138
QP[N]	Position	141

Command	Description	Page
QT[N]	Time	141
QV[N]	Velocity	141
RM	Reference mode: external (analog) referencing enabled/disabled	145
TR	Target radius	164
UM	Unit mode: stepper, torque control, speed control position control or dual loop	170
VH[N]	High reference limit	174
VL[N]	Low reference limit	174
XM[N]	X Modulo	184
YM[N]	Y Modulo	188

2.6 Communication Commands

Command	Description	Page
PP[N]	Define the parameters of the CAN or RS-232 communication	132

2.7 Control Filter Commands

Command	Description	Page
GS[N]	Gain scheduling	65
KG[N]	Gain scheduled controller parameters	89
KI[N]	PID integral terms array	90
KP[N]	PID proportional terms array	90
KV[N]	Advanced filter for speed loop	92
XA[N]	Extra parameters (more)	182

2.8 Protection Commands

Command	Description	Page
ER[N]	Maximum tracking errors	61
HL[N]	Over-speed limit and position range limit	66
LL[N]	Low actual feedback limit	97

PL[N]	Peak current, in amperes; and peak duration, in seconds	127

2.9 Data Recording Commands

Command	Description	Page
BH	Get a sample signal as hexadecimal	22
RC	Variables to record (two variables at each recording sequence)	142
RG	Recording gap, in samples. Gap between consecutive data recordings.	143
RL	Record length	144
RP[N]	Recorder parameters	146
RR	Recording on/off	148
RV[N]	Recorded variables	151
YM[N]	Auxiliary sensor modulo count	188

2.10 User Program Commands

Command	Description	Page
CC	Compile program	32
CP	Clear application program	37
DL	Receive a program downloaded from host computer to Metronome. Can be used only in Composer software.	41
HP	Halt program execution	70
KL	Kill motion and stop program (like HP)	91
LP[N]	List parameters	98
LS	List program	99
MI	Mask interrupt	105
PS	Program status	135
XC	Continue program execution from current pointer, optionally until a given breakpoint	183
XQ	Execute program, optionally starting at a given label and until a given breakpoint	183

2.11 General Commands

Command	Description	Page
AR[N]	Active recording session - Special Wizard commands	19
CD	CPU dump: CPU and database exception summary	33
DF	Download firmware	40
EO	Echo mode	60
HX	Select hexadecimal or decimal mode	72
LD	Load parameters form flash memory	95
PN[N]	Integer parameters	130
RS	Reset Metronome to a pre-defined state and parameter value	150
SI	Synchronize correction tables	153
SV	Save parameters to flash memory	160
TM	System time	163
TT[N]	Sampling time of motion controller	165
TV[N]	Correction tables data	166
TW[N]	Wizard command	167
UF[N]	User float array	168
UI[N]	User integer	169
VC[N]	Phase voltage command - Special wizard commands	172
WI[N]	Metronome data, mainly for use by Composer	177
WS[N]	Metronome data, mainly for use by Composer	179
ZX[N]	User program and auto-tuning temporary storage	190

Chapter 3: Alphabetical Listing

This chapter lists all the commands in alphabetical order, along with detailed definitions and examples of each command.

The description of each command includes the following items:

Purpose: The operation or task of the command

Attributes: The characteristics of the command

Type: One of the following:

- A **command**: An instruction to do something. For example, the BG (Begin Motion) command starts a new motion profile.
- A **parameter**: A data item that may be used later. For example, the AC (Acceleration) parameter is required for calculating subsequent motions.
- A **status report**: Get a value, such as the motor speed, a digital input or the reason for the last motor failure.

The parameters and certain commands have numerical values, as follows:

- **Integer**: A 32-bit long integer
- **Real**: A 32-bit floating point number (IEEE style)
- **String**: A set of printable ASCII characters

Integer variables may have the following attributes:

- **Bit field**: The integer should be understood not as a number but rather as a combination of binary fields. For example, the IP (Digital Input) command reads many On/Off switches to the same integer, allocating one bit for each.
- **Option**: A selector that may accept one of several options. For example, the motor direction may be set to forward or reverse, symbolized by the numbers 0 and 1 respectively.

Source: Defines the "agents" that may use the command, as follows:

- RS-232 communication
- CANopen communication
- User program

> The command access rights are not equal for all sources. For example, CANopen binary interpreter cannot use the string commands listed in this manual. Another example is the XQ (Execute Program) command that, of course, cannot be performed from within a program.

Restrictions: The use of certain commands is illegal in certain contexts. The reasons for this may be:

- *Safety:* For example, it is not safe to change the direction of the feedback while the motor is running.
- *Relevance:* For example, a torque command cannot be specified in speed control mode (UM=2); in speed mode, the drive automatically sets the torque.
- *Consistency:* A parameter may be inconsistent with the specification of other parameters. For example, in point-to-point mode, the position absolute value (PA) may be no higher than the maximum allowed position reference (VH[3]).

Default values: Default value and storage type.
Volatile variables are reset to their defaults with each power on.
Non-volatile variables can be stored using the SV command.
Stored non-volatile values are read from storage upon power on and can be reset to their defaults using the RS command.

Range: Range definition: For example, the position command may be specified in the range [-1,073,741,824…-1,073,741,823]

Unit mode (UM): Defines the function of the drive. The unit modes are:

- UM=1 Torque control
- UM=2 Speed control
- UM=3 Micro-stepping
- UM=4 Dual-feedback position control
- UM=5 Single-feedback position control

Activation: Specifies when the entered parameter value should be used. Activation may be:

- Immediate As soon as the command is processed
- Triggered By another command

For example, the AC (acceleration) parameter should only affect the next motion, triggered by the BG command.

Examples: Simple examples of the command usage. All examples are given in RS-232 syntax.

See also: Related commands

Reference chapter in the *SimpIIQ for* Chapter or section that contains relevant details pertinent to the command.

Limit Ranges

The following table lists the value ranges for defining the limits of the system.

Subject	Values
Position counter ranges	Main position counter is subjected to a modulo counting with the following ranges: XM[1]: Lowest value XM[2]: Highest value Range: [-5x10^8…5x10^8] counts Auxiliary position counter is limited to: YM[1]: Lowest value YM[2]: Highest value Range: [-5x10^8…5x10^8] counts
Velocity range	Range for **Quadrature Encoder**: [-40,000,000…40,000,000] counts/sec Range for **other feedbacks**: [-80,000,000…80,000,000] counts/sec EF[1]: Filter for main velocity sensor EF[2]: Filter for auxiliary velocity sensor
Acceleration/Deceleration ranges	Range: [100…1,000,000,000]
Stop deceleration range	Range: [100…1,000,000,000]
Torque limits	Range of torque command is subjected to the following limits: CL[1], PL[1] Range: [-MC…MC] Note: ▪ The multiplication of the PWM frequency reduces the torque limit.

Table 3-1: **Limit Ranges**

AB[N] – Absolute Encoder Setting Parameters

Purpose:

This command is reserved for future hardware that will support an absolute encoder.

AC - Acceleration

Purpose:
Defines the maximum acceleration in counts/second^2. This parameter is used in speed mode (UM=2) and position control modes (UM=3, 4, 5) in PTP (PA, PR) and jogging (JV) reference modes.

The AC parameter does not affect the present motion. It is used for planning the next motion, which is initiated by a BG command.

If AC is smaller than SD, the maximum possible acceleration will be limited to SD and the value of AC will be ineffective.

Attributes:	**Type:**	Parameters, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	20,000,000 (RS), Non-volatile
	Range:	Acceleration range
	Unit modes:	UM=2, 3, 4, 5
	Activation:	BG
	SimplIQ:	Similar

Typical applications:
1. Define acceleration limits for the motion (UM=2)
2. Plan a profiled motion (UM=3, 4, 5)

See also:
DC, SP, SV, PA, PR, BG

Application notes:
The Position Reference Generator

The speed software command

AG[N] - Analog Gains Array

Purpose:
Sets the gains for preconditioning analog signals, when RM = 1:

- AG[1] sets the gain of analog input #1 when used as a torque command (UM=1, 3).
- AG[2] sets the gain of analog input #1 when used as a speed command (UM=2).

When RM = 0, the AG[N] parameters are ignored.

The meaning of the analog gains depends on the unit mode, as shown in the following table.

Value	Description	Units
UM=1, 3	One volt at the analog reference input command controls the motor phase current of AG[1] amperes.	Ampere/volt
UM=2	One volt at the analog reference input command controls a speed of AG[2] counts/second.	Count/second/volt

Table 3-2: **Analog Gains - Analog Input #1**

Notes:

- AG[1] defines motor phase amperes and not RMS amperes.
- In stepper mode (UM=3), the two external inputs play different roles: The analog input voltage sets the motor power while the follow pulse/direction or quadrature input determines the position.
- The polarity of the analog reference signal may be reversed by setting the sign of the AG[N] parameter accordingly.

Attributes:	**Type:**	Parameter, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	AG[1]=0.1 AG[2]=1 Non-volatile
	Range:	AG[1]: [-10000…10000] AG[2]: [-20,000,000…20,000,000]
	Index range:	[1, 2]
	Unit modes:	All
	SimplIQ:	Similar, ranges changed
	Activation:	Immediate

See also:
AN[N], AS[N], UM, RM, VH[N], VL[N]

Application notes:
Speed reference generation
Open loop stepper control

AN[N] – Analog input array

Purpose:

- AN[1] reports the analog input #1 value after offset correction (AS[1]), in volts.
- AN[2] reports the analog input #2 value after offset correction (AS[2]) in volts, for products that support a second analog input.
- AN[3] … AN[5] – see table below.
- AN[6] reports the DC line voltage value, in volts. Note that for AC products, this value is after rectification. For example, with 1-phase 220 VAC supply, expect AN[6]=√2x220=310 V. with 3-phase 380 VAC L-L supply, expect AN[6]=√2x380=540 V.
- AN[7] reports the duty cycle value of the PWM signal after offset correction in fractional units.

Attributes:

Type:	Status report, Real
Source:	Program, RS-232, CANopen
Restrictions:	None
Unit modes:	All
SimplIQ:	Modified

AN[3..5] read motor phase currents, in Amperes. The meaning depends on the motor type

	2-phase (stepper)
AN[3]	Phase A
AN[4]	Phase B
AN[5]	0

Typical applications:
1. Reading external sensors that provide ±10 V.
2. Reading analog or PWM references for either velocity or current.
3. Reading phase currents and line voltage.

Notes:

- Analog input #1 serves as reference input for analog torque or analog velocity command while in auxiliary reference mode (RM=1).
- Different SimplIQ drives support a different number of analog inputs. For specific details consult the drive's *Installation Guide*.

See also:
AG[N], AS[N], PW[N]

AR[N] – Active Route array

Purpose:

This command determines that some commands will be routed to the motion processor instead of the standard processing.

- AR[1] =1 sets the recorder to record directly from the motion processor.
- AN[2]=1 sets the profiler to work directly from the motion processor.

Attributes:		
	Type:	Parameter, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	0 (RS), volatile
	Range:	0 or 1
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Absent

Typical applications:
Tuning of the motion control parameters at a higher time resolution than is available for the SimplIQ processor.


Under most circumstances, this command is used only by the tuning environment.


Note:

- If you terminate a tuning environment session abnormally, the drive may remain with a non-default AR[N] setting. This may cause abnormal drive behavior. When a tuning environment session aborts abnormally, it is recommended to reset the drive by power down.

Application notes:

The recorder

AS[N] - Analog Input Offsets Array

Purpose:
Compensates for offsets of the analog signals, which may be caused by the limited precision of the *SimpIIQ* electronics.

At times, the signals at the A/D converter may be offset: that is, the A/D reading may be non-zero when a zero reading is desired. This offset may disturb normal operation. An offset reference or feedback signal may cause a motor to "crawl" when a complete stop is desired.

The analog offset subtracts from the analog input as follows:
Corrected signal = A/D reading – Analog offset

- AS[1] - Analog input offset command for analog input #1, in volts
- AS[2] - Analog input offset command for analog input #2, in volts

Attributes:		
	Type:	Parameter, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	0 (RS), Non-volatile
	Range:	[-10.0…10.0] *5 mV resolution*
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

Notes:

- To null the input offsets of the drive, short the analog inputs to ground. Then set AS[1] = AN[1] and AS[2] = AN[2] for analog input #1 and #2 respectively.
- Each of Elmo's *SimpIIQ* drives support a different number of analog inputs. For specific details consult the drive's *Installation Guide*.

See also:
AG[N], AN[N]

Application notes:
Speed reference generation
Open loop stepper control

BG - Begin Motion

Purpose:
Immediately starts the next programmed motion.

- In ***software speed mode*** (UM=2), BG activates the latest JV, and also the new smooth factor (SF), acceleration (AC) and deceleration (DC).
- In ***stepper or position mode*** (UM=3, 4 or 5), BG starts the latest position mode programmed: a point-to-point motion (PA), a jogging motion (JV) or any type of tabulated motion (PVT or PT).

Each motion mode starts with its entire set of parameters. For example, starting a point-to-point motion activates the present value of acceleration (AC), deceleration (DC), smooth factor (SF) and speed (SP).

The BG command may be used to *modify* the parameters of the present mode, and not only to program new modes. For example, a BG command in point-to-point mode modifies the active AC parameters (and all other active motion parameters) with its last programmed value.

A "hardware BG" can be accepted via one of the digital inputs (refer to the IL[N] command).

Attributes:	**Type:**	Command, No value
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=1
	Unit modes:	UM=2, 3, 4, 5
	Activation:	Immediate
	SimplIQ:	Similar

Notes:

- In position mode (UM=3, 4, 5), BG does nothing if a motion mode (JV, PA, PV, PT) was not previously submitted.
- In "Quick stop" mode (refer to the Elmo *CANopen Implementation Guide*), BG is blocked and returns an error. "Quick stop" mode can be command controlled by a CAN master using the DS402 standard control word (object 0x6040).

See also:
IL[N], BT

Application notes:
The Position Reference Generator

The speed software command

BH - Get a Single Recorded Signal as Hexadecimal

Purpose:
Uploads the values recorded by the recorder to a host. In response to the BH command, the sends a recorder data message is Hexa-binary form (A binary stream represented by printable characters). Although recorded data transmissions may take long time, the drive can accept new commands and run user program code while transmitting the recorded data.

Attributes:	**Type:**	Command, Integer, Bit-field
	Source:	RS-232
	Restrictions:	RR=0 (valid recorder data is ready), Not while executing a previously-requested BH=n command
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar


Under most circumstances, this command is used only by the tuning environment.


The BH command is designed to optimize data transfer from the drive to the host, assuming that the host has the computing power to analyze the drive message.

See also:
AR[N]

Application note:
The Recorder

BP[N] - Brake Parameter

Purpose:
Defines the timing of the brake system in the motor when at least one of the digital outputs has been defined by the OL[N] command as a brake. *For safety reasons, a brake-active output releases the brake so that the brake is activated when the drive is not powered on. The brake output is always defined as active low.*

When the brake is released at motor start (MO=1), the drive allows the brake time to disengage before motion begins. During this time, the drive keeps the motor in its starting position. When the motor is turned off (MO=0), the drive first commands the brake to engage. Then, for a time, it keeps the motor in place while the brake actually engages.

- BP[1] - Defines the delay for engaging the brake after the motor is disabled (msec).
- BP[2] - Defines the delay required to disengage the brake after the motor is enabled (msec).

Notes:

- If the motor is disabled by an emergency in real time, the brake is activated at the instant the motor is disabled. The motor freewheels until the brake is fully engaged.
- Response time to interpreter commands (from the user program or communication) is extended during motor disable (MO=0) and enable (MO=1) in BP[1] and BP[2] milliseconds respectively.
- Automatic phasing with oscillation (CA[17]=3) is not impossible for a system that requires brake activation.

Attributes:		
	Type:	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0
	Default values:	BP[1]=0 BP[2]=0 Non-volatile
	Range:	BP[1]: [0…500] BP[2]: [0…500]
	Index range:	[1, 2]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

See also:
OL[N], OP

BT - Begin Motion at Defined Time

Purpose:
Starts motion at the defined time. This command is designed to start the simultaneous motion of several axes. It is similar to the BG command with the following difference: BG starts motion immediately whereas BT begins at the defined time.

Syntax:
BT=N
where N is the absolute time in microseconds

 **Note:**

An immediate Begin (BG) overrides a pending BT. Motion starts immediately, and the BT is canceled.

Attributes:	**Type:**	Command, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=1
	Unit modes:	UM=2, 3, 4, 5
	SimplIQ:	Similar

See also:
BG, TM

BV - Maximum Motor DC Voltage

Purpose:
Reports the maximum DC drive bus voltage, in volts. This command reads the rating of the power amplifier hardware.

 **Notes:**

It is not recommended to work with DC voltage in excess of 90% BV. At about 95%BV, over-voltage shall trip.

Attributes:	**Type:**	Status report, Integer
	Scope:	Program, RS-232, CANopen
	Restrictions:	None
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

See also:
MC

CA[N] - Commutation Array

Purpose:
Defines motor and commutation parameters. The CA[N] array includes the parameters of the initial motor setup. The CA parameters need to be well defined in order to ensure that the motor rotates efficiently, or even rotates at all. CA[N] also sets the feedback direction.


Under most circumstances, this command is used by the tuning environment only.

It is not recommended to modify these parameters manually.


Notes:

Normally, for brushless and stepper motors, a commutation search (identifying the electrical rotor position with respect to the stator position) is made once at the first Motor On. When any of CA[N] is set, the transformation angle resets and the next Motor On will initiate a new commutation search.

The parameters in the following tables define the location and polarity of the Hall sensors and encoder.

Command	Description
CA[1]	Digital Hall sensor A polarity (1 for active high, 0 for active low).
CA[2]	Digital Hall sensor B polarity (1 for active high, 0 for active low).
CA[3]	Digital Hall sensor C polarity (1 for active high, 0 for active low).
CA[4]	Actual Hall sensor connector to Hall A connector pin: 1 for A, 2 for B and 3 for C.
CA[5]	Actual Hall sensor connector to Hall B connector pin: 1 for A, 2 for B and 3 for C.
CA[6]	Actual Hall sensor connector to Hall C connector pin: 1 for A, 2 for B and 3 for C.
CA[7]	Offset of digital Hall sensors. The units are 1024/electrical revolution.

Table 3-3: **CA Vector - Digital Hall Sensor Parameters**

The parameters in the table that follow are required for the Analog Encoder, Resolver or Analog Halls signal scaling:

Command	Description
CA[9]	Relative phase of the Analog Encoder sinusoidal signals (or Analog Halls or Absolute Coarse/Fine Encoder(fine mode)), in 65,536 units per cycle. In most systems, CA[9] will fall in the range of [-2048…2048].

Command	Description
CA[10]	Resolver or Analog Halls offset – the value of the analog sensor readout at the electrical zero of the motor
CA[11]	Offset for the A (sine) channel of the *Analog Encoder*, *Resolver*, *Analog Halls or Absolute Coarse/Fine Encoder(fine mode)*. The offset is given in ADC units in the range of [-4500…4500].
CA[12]	Offset for the B (cosine) channel of the *Analog Encoder*, *Resolver*, *Analog Halls or Absolute Coarse/Fine Encoder(fine mode)*. The offset is given in ADC units in the range of [-4500…4500].
CA[13]	Relative gain of the A (sine) channel of the *Analog Encoder*, *Resolver*, *Analog Halls or Absolute Coarse/Fine Encoder(fine mode)* with respect to the B (cosine) channel in the range of [20000…40000].

Table 3-4: **CA - Analog Feedback Scaling**

Command	Description
CA[16]	Feedback direction: 0: Use feedback reading as is 1: Invert the direction of the feedback reading Changing CA[16] does not affect the present position count. Direction changes only when counting future feedback pulses.
CA[17]	Commutation method. 0: Use only Hall sensors. The electrical angle resolution will be 60 electrical degrees. 1: Hall sensors are used together with a high resolution sensor. On the first Hall sensor transition, the exact electrical angle is locked and commutation continues by position sensor increments. The Hall sensors continue to monitor commutation correctness – refer to PN[1]. 2: Halls are sole commutation source. On high speeds, use time-based interpolation for smoothing the electrical angle estimate. 3: Commutation initializes by auto phasing, in the oscillation method (see CA[15],CA[24],CA[26],CA[27]). After initialization, commutation continues by position sensor increments. 4: Commutation initializes by auto phasing, in the magnetic saturation method (see CA[15],CA[24],CA[26],CA[27]). After initialization, commutation continues by position sensor increments. 5..7: Reserved

Command	Description
	8: Every Hall sensor transition: the exact electrical angle is locked and commutation continues by position sensor increments.
CA[18]	Feedback bits ("counts") per revolution, after resolution is multiplied by 4, in the range [6…530,000,000]. For **Analog Encoders**, the resolution for commutation is always taken as $x2^{16}$, regardless of CA[31]. For Analog Hall sensors always use CA[18]=65536 For systems with **Halls only**, CA[18] is calculated as CA[19] * 6
CA[19]	The number of feedback counts per electrical revolution is CA[18]/CA[19]. For brushless motors, this is the number of pole pairs. For steppers, this is one quarter of the steps count.

Table 3-5: **CA Vector - Feedback Setup Parameters**

The parameters in the tables that follow are required for commutation setup.

Command	Description
CA[20]	Digital Hall sensors present: 0: No digital Hall sensors are connected. If the commutation angle is not yet known, then at motor on a commutation search will be made. No digital Hall input consistency checks will be made. 1: Digital Hall sensors are connected. Upon motor on, commutation will be performed according to the digital Hall sensors. Continuous encoder-based commutation will begin when the first Hall edge is identified. The drive performs commutation checks by continuously comparing the encoder-derived commutation angle with the Hall sensor recorded status.
CA[21]	Position sensor present: 0: Ignore the position sensor inputs. Commutation will be based on the digital Hall sensors only. 1: The position sensor will be used for commutation.
CA[22]	Main feedback type: 0-1: Reserved. 2: Quadrature incremental encoder signals. 3: Analog encoder with 1v p-p sine\cosine signal. 4: Reserved 5: Reserved

Command	Description
	6: Main feedback entry used as input for digital halls signals 7-11: Reserved
CA[25]	Motor direction: 0: Keep the original motor direction, as connected by user. 1: Reverse phase driving. The effect of CA[25] is to set the direction of torque/force for positive current demand.
CA[28]	DC motor: 0: Three-phase brushless motor, Uses the M1, M2, and M3 motor terminals. 1: DC motor — do not perform commutation. Current will flow continuously from the M1 motor connector pin of the servo drive to the M2 terminal. The C terminal conducts no current. 2: Two phase motor (stepper). One phase is connected between the M1 and M2 terminals, and the other phase is connected between M3 and M4

Table 3-6: **CA Vector - Commutation Setup Parameters**

Command	Description
CA[15]	Signal frequency for "No Hall" commutation search process. This signal is derived from the sampling time, according to the following formula: Time = 128 * TT[1] * $2^{CA[15]}$ * 1e-6. The frequency is 1/Time Hz.
CA[24]	The minimum motor movement to perform result analysis, in counts. When this variable is too low, the commutation search process might fail (MF=0x10,000).
CA[26]	Starting torque for motor-on commutation search process in percentages. Starting torque = (CA[26]/100))*CL[1]
CA[27]	Maximum acceptable number of iterations for auto-phasing process. If the process fails due to overload (motion amplitude is less than CA[24]), the auto-phasing may be repeated CA[27] times, with the current being doubled at every iteration. If the peak current (PL[1]) is reached at any attempt, the auto-phasing process will stop even if CA[27] allows more iterations.

Table 3-7: **CA Vector - Automatic Commutation Search Parameters, no Hall Sensors**

Command	Description
CA[23]	Counts per meter (any positive integer): 0: Rotary motor 1: Counts per meter in a linear motor This parameter is not used directly by the drive but is rather stored there for the convenience of the host.

Table 3-8: **CA Vector - Miscellaneous Parameters**

Command	Description
CA[31]	Resolution for one cycle of the analog signal. Analog encoder cycles at one revolution x $2^{CA[31]}$ counts/rev. CA[31] is in the range [2..12]. Changing CA[31] resets the position counter. For linear motors the resolution is per meter. CA[31] does not affect commutation.

Table 3-9: **CA - Resolution of the Analog Encoder**

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0
	Default values:	CA[1]=CA[2]=CA[3]=1,CA[4]=3, CA[5]=2, CA[6]=1, CA[13]=32768, CA[15]=4, CA[16]=1, CA[17]=1,CA[18]=4096, CA[19]=2, CA[20]=1, CA[21]=1, CA[22]=2, CA[24]=5, CA[25]=1, CA[26]=20, CA[29]=1 All other CA parameters are 0 (RS), Non-volatile
	Range:	As defined in the previous tables
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Modified, see notes below

Notes:

- The CA parameters are usually set automatically by the tuning utility programs. Avoid setting the CA[N] parameters manually unless you are sure of what you are doing.
- Unused indices are reserved for compatibility with other drive models.

- Main changes from SimplIQ:
  - CA[7] units changed from encoder units to 1024/rev.
  - CA[17] newly defined, now determines exactly commutation method.
  - CA[18]: For analog encoder, the interpolation ratio is fixed, regardless of CA[31].
  - CA[28] allows definition of 2-phase (stepper) motors.
  - For Analog encoder, commutation is made with the full resolution interpolated position. CA[31] does not affect commutation.

See also:
MO, UM

Application notes:

CA[1]…CA[7],CA[17],CA[20]: Hall Sensors

CA[16],CA[17],CA[18],CA[19],CA[21]: High resolution sensors for commutation

CA[25]: Commutation

CA[15],CA[24],CA[26],CA[27]: Auto phasing by oscillation

CC - Compiled Program Ready

Purpose:
Serves as the last stage of the user program downloading process, verifying the downloaded user program checksum and marking it "ready for use."

The CC=N command specifies the program checksum. If this value coincides with the actual program checksum, the "Program ready" internal flag is set on. Otherwise, an error is returned. The CC query returns 0 if no active program is present, and 1 if the "Program ready" internal flag is set on.

Attributes:	**Type:**	Parameter, Integer
	Source:	RS-232, CANopen
	Restrictions:	MO=0, Program not running
	Range:	[0…2^{32}]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar


Under most circumstances, this command is used only by the Elmo Studio.


The execution of a CC=N command may take a significant amount of time, approximately 1 second.

CD - CPU Dump

Purpose:
Returns the status of the CPU and the database. Call CD if:

- The SR report indicates a CPU exception.
- The MF report indicates a CPU exception.
- An attempt to start the motor returns a "Bad database" error code.

The CD report returns a string similar to the following:
Null Address=0
Failure Address=0
Called Handler=none
Database Status:
Database OK
Tables loaded to motion processor ok
Motion processor database ok

where:

- "Null Address" is the code address at which a CPU exception occurred. A "0" indicates a normal condition.
- "Failure Address" is the code address at which a stack overflow has occurred. A "0" indicates a normal condition.
- "Called Handler" is the type of CPU exception that occurred. A "none" indicates a normal condition.
- "Database Status" indicates if the recent database check at MO=1 — at power up or during a save (SV) — revealed a consistent database. "Database OK" indicates the normal condition.
- "Tables loaded to motion processor" indicates the synchronization between the tables data stored in the SimplIQ processor and the motion processor – see the SI command.
- "Motion processor database" may indicate conflicts in data items passed to the motion processor.

Attributes:	**Type:**	Status report, String
	Source:	RS-232
	Restrictions:	None
	Unit modes:	All
	SimplIQ:	Modified – see notes

Notes:

- SimplIQ for Steppers difference: added motion tables and motion CPU status.
- If an LD command fails, CD reports the reason for the failure by adding the string "Couldn't load from serial flash" followed by the reason for the failure.

Example:
Null Address=0
Failure Address=0
Called Handler=none
Database Status:
CA[4], error code=37

Motion processor database

This CD report indicates that the database is inconsistent because two of the parameters CA[4], CA[5] and CA[6] are equal.

See also:
MF, SR, MO, EC

CL[N] - Current Continuous Limitations and Motor Stuck Protection Parameters

Purpose:
Defines the continuous loading of the drive.

- CL[1] defines the maximum allowed continuous motor phase current, in amperes. This parameter is used to protect the motor from over-current, and the load from excessive torques. The motor current (torque) command is normally limited to its peak limit, as defined by PL[1]. After a short period of torque demands higher than CL[1] (as defined by the PL[2] parameter and equations in the *SimplIQ for Steppers Application Note*), the torque command limit is decreased to CL[1]. The torque command remains limited to CL[1] until the average torque demand falls below 90% of CL[1] for a few seconds. CL[1] has no effect if CL[1] > PL[1].
- CL[2] and CL[3] define how the motor stuck protection is handled. A stuck motor is a motor that does not respond to the applied current command, due to failure of the motor, the drive system or the motion sensor.
  CL[2] defines the tested torque level as a percentage of continuous current limit CL[1].
  CL[3] states the absolute threshold of the main sensor speed under which the motor is considered to be not moving. If the motor is stuck, motion fault MF=2,097,152 (0x200,000) is set.
  If CL[2] < 2, the motor stuck protection is not applied.
  For other values of CL[2], the motor is disabled and MF is set to 0x200000 if the motor current command level exceeds a selected level for more than 3 seconds, without the result of a significant motor speed, as defined by CL[3].

Attributes:		
	Type:	Parameter, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	CL[1]=0 (RS), Non-volatile, CL[2]=0 (RS), Non-volatile, CL[3]=60 (RS), Non-volatile
	Range:	CL[1]: [0…MC/2] CL[2]: [0…100] CL[3]: [0…16,000]
	Index range:	[1…3]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

Example:
If CL[2]=50 and CL[3]=500, the drive will abort (reset to MO=0) with motion fault (MF) 0x200000 if the torque level is kept at least 50% of the continuous current, while the absolute value of the shaft speed does not exceed 500 counts/sec for a continuous 3 seconds.

Notes:

- The motor stuck protection is always applied to the main sensor. In dual loop applications, this protection does not pertain to failures in the auxiliary sensor.
- The time constant of 3 seconds is taken because almost every motion system applies high torques for short acceleration periods while the speed is slow.
- The minimum current limit is MC/128. If CL[1] < MC/128, the CL[1] value will be accepted, but the actual current value will be limited to MC/128.

See also:
LC, MC, PL[N], TC, MF

Application notes:

CL[1]: Current limiting

CL[2], CL[3]: Motor does not move protection

CP - Clear Program

Purpose:
Clears the entire user area in the serial flash memory. The CP instruction must be used before any attempt to write a new program to the drive.

Attributes:	**Type:**	Command, No value
	Source:	RS-232, CANopen
	Restrictions:	MO=0, Program isn't running
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar


Under most circumstances, this command is used only by the Elmo Studio.


Notes:

- CP command execution may take a significant amount of time.
- CP deletes the user program permanently.
- Downloading a new program without prior use of CP can cause undefined behavior.

See also: SimplIQ Programming and Language Guide

DC - Deceleration

Purpose:
Defines the maximum deceleration in counts/second^2. This parameter is used in speed mode (UM=2) and position control modes (UM=3, 4, 5) in PTP (PA, PR) and jogging (JV) reference modes.

The AC parameter does not affect the present motion. It is used for planning the next motion, which is initiated by a BG command.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	20,000,000 (RS), Non-volatile
	Range:	Acceleration range
	Unit modes:	UM=2, 3, 4, 5
	Activation:	BG
	SimplIQ:	Similar

Notes: The DC parameter does not affect profile termination by switch events or by the ST command. For profile termination, refer to SD.

Typical applications:
1. Define acceleration limits for the motion (UM=2)
2. Plan a profiled motion (UM=3, 4, 5)

See also:
AC, SP, SV, PA, PR, BG, SD

Application notes:
The Position Reference Generator

The speed software command

DD - CAN Controller Status

Purpose:
Returns the status of the CAN controller as a string in hexadecimal form without a "0x" prefix. DD is valid only for drives that support CAN controllers.

Call DD if you:

- Suspect that the CAN controller is in Bus Off (no communication) mode.
- Suspect that there are many error frames on the CAN bus.
- Wish to monitor the CAN controller error activities.

The DD command reflects object 0x2082 (refer to the Elmo *CANopen Implementation Guide* for more information).

DD value reports:

- CAN receiver flag, indicating the following states:
  - Overrun
  - Bus off
  - Transmitter error
  - Receiver error
  - Transmitter warning
  - Receiver warning
- CAN receive error counter, reflecting the status of the MSCAN receive error counter.
- CAN transmit error counter, reflecting the status of the MSCAN receive error counter.
- Network status, which may be one of the following values:
  - 1: Disconnected
  - 2: Connected
  - 3: Preparing
  - 4: Stopped
  - 5: Operational
  - 127: Pre-operational

All data is received from the hardware.

Attributes:		
	Type:	Status report, String
	Overloaded:	No
	Source:	RS-232, CANopen
	Restrictions:	None
	Unit modes:	All
	SimplIQ:	Similar

DF/DT - Download Firmware

Purpose:
Downloads a new firmware version. These commands are designed as part of a sequence that is normally performed and controlled by the Composer program, which reads the firmware update file provided by Elmo, and performs a sequence of actions that includes the DF/DS command.


This command is used only by the Elmo Composer.


Notes:

- After new firmware is downloaded, the drive reboots. All data stored in temporary variables in the RAM is lost.
- Loading new firmware does not normally affect the non-volatile application variables in the data flash memory. Downloading a major software revision may, however, destroy the non-volatile data. The Composer program will warn you of risks of non-volatile data losses.
- The DT command is used for downloading the new firmware version to the motion control processor.
- SimplIQ for Steppers must use an updated Composer program that correctly identifies SimplIQ for Steppers.

Attributes:	**Type:**	Command
	Source:	RS-232 only
	Restrictions:	MO=0, Program not running
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Modified – See notes

DL - Download Program

Purpose:
Downloads data to the non-volatile memory of the drive. The DL command is used primarily to download compiled user programs to the drive.

The format of a DL command is:
DL##[hex binary data][esc]checksum]

Attributes:	**Type:**	Command, String
	Source:	RS-232
	Restrictions:	MO=0, Program not running
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar


Under most circumstances, this command is used only by the Elmo Studio.


Notes:

- The start memory address for downloading is defined by the LP[1] command.
- Each data payload is terminated by the 16-bit checksum for the send message.
- The DL command automatically clears the Program Ready internal flag.
- The DL command takes time to execute because it needs to burn flash and verify.

See also:
LS, CC, LP[N]

SimplIQ Programming and Language Guide

DV[N] - Reference Desired Value

Purpose:
Reports the reference commands to the position, speed and current controllers of the drive. DV[N] reports the final value of the controller references, as synthesized by all their sources: software reference generators, external reference inputs and external control loops.

- **For UM=1**, DV[1] reports the torque command. DV[2] and DV[3] report zero.
- **For UM=2**, DV[2] reports the speed command. DV[1] reports the torque command derived by the speed controller. The speed command may differ from the desired speed as specified by the JV command and the analog input, because the filters and the profiler "smooth the edges" before transferring the desired speed as a speed command to the speed controller. DV[3] returns zero.

  The speed command reported by DV[2] is generated by the sum of an external analog reference and a software reference. This sum is further processed for acceleration and speed limiting as well as the switch response. DV[4] and DV[5] retrieve the external and software components of DV[2].
- **For UM=3, 4, 5**, DV[1] reports the torque command derived by the speed controller, DV[3] reports the command to the position controller and DV[2] reports the speed command, which is the rate of change of DV[3]. The position command may differ from the desired position as specified by software commands and by superimposed analog input, because the stop manager may affect the position command to the controller.

  The position command reported by DV[3] is generated by the sum of an external follower reference and a software reference. This sum is further processed for acceleration speed limits, as well as the switch response. DV[6] and DV[7] retrieve the external and the software components of DV[3].
- **In summary**:
  DV[1] reports the current command value.
  DV[2] reports the velocity command value.
  DV[3] reports the position command value.
  DV[4] reports the external speed command (0 for RM=0).
  DV[5] reports the software speed command.
  DV[6] reports the external position command (0 for RM=0).
  DV[7] reports the software position command.

When the motor is disabled (MO=0), DV[N] returns zero.

Attributes:		
	Type:	Status report, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Index range:	[1…7]
	Unit modes:	All
	SimplIQ:	Similar

See also:

AR[N]

Application note:
Unit Modes

Note:

- If AR[2]=1, The profiled motion is generated entirely within the motion processor and the DV command returns values directly from the motion processor.

EC - Error Code

Purpose:
Reports the processing status of the last accepted command that returned an error.

Notes:

- When the processing of a command fails, the error code is returned immediately with a question mark in the response to that command. The error code returned with the command response is binary, so it may not be easily seen. The EC command returns a printable (ASCII) value of the error code.
- The EC command cannot be used reliably from the Composer Smart Terminal because the Composer generates continuous communication with the servo drive. The returned EC value probably reflects the status of the latest Composer command, not the status of the last Smart Terminal command.
- New error codes may be defined as software evolves.
- The SimplIQ drive stores internally its entire set of error code and a short comment describing each, so the Composer program may correctly interpret errors from any software version.

The following table lists the error codes reflected by the EC command.

Error Code	Error String / Description	Example / Remedy
2	Bad command. The interpreter has not understood the command.	XF=2; is an error because there is no XF command. MC=2; is an error because the MC cannot be changed by the interpreter.
3	Bad index. Attempt to access a vectored variable out of its boundaries.	DV[6] is an error because the index range is 1 - 3. Observe the index range for the used command.
5	Has no interpreter meaning. An unrecognized character has been found where a command was expected.	A*=3 is an error because a command mnemonic consisting of two alphabetic characters was expected.
6	Program is not running.	This command requires a running program.

Error Code	Error String / Description	Example / Remedy
7	Mode cannot be started - bad initialization data.	This error is returned when preset values of a function are wrong. For example, there may be a conflict between the first index in the PVT table (PV) and the available write pointer (MP[6]) when PVT motion begins.
11	Cannot write to flash memory. An error interfacing the serial flash has occurred.	Most probably a hardware problem.
12	Command not available in this unit mode.	PA=1000 is an error if UM=2, because the position command cannot be given in this mode.
13	Cannot reset communication – UART is busy. Modification of the parameters of the serial communication has been attempted while the communication line is busy.	
18	Empty assign. The right side of an equation is missing.	AC=; is an error because the interpreter expects a numerical value to appear after the = sign.
19	Bad command format. An unresolved syntax error in the command has occurred.	Refer to this manual for the correct command syntax.
21	Operand out of range. Assignment of an illegal value to a parameter has been attempted.	JV=100,000,000 returns this error because the required speed is beyond the limits of the drive.
22	Zero division.	JV=0; PX=1000/JV returns this error.
23	Command cannot be assigned.	BG=3000 returns this error because BG is an execution command that does not have a value.
24	Bad operator. An unrecognized character has been found in an expression where an operator has been expected.	IA[1]=3$VX is an error because $ is not a recognized operator.

Error Code	Error String / Description	Example / Remedy
25	Command not valid while moving.	PV=n while in PVT motion is an error because the PV=n command sets the read pointer of the PVT table manually; this pointer is set automatically in PVT mode.
26	Motion mode not valid. A Begin Motion was attempted but the parameters of the motion were not properly set.	PV=n; BG is an error if the first valid line of the PVT table is smaller than the last valid line.
28	Out of limit range. A command was specified out of its permitted limits.	VH[2]=1000; SP=2000 is an error because the latter command specifies that point-to-point motions should reach the speed of 2000 counts/sec, whereas the first command limits the maximum speed command to 1000.
30	No program to continue. An XC command has been issued but there is no halted program to continue.	
32	Communication overrun, parity, noise or framing error.	Ensure that communication lines are well connected with adequate ground, and that the baud rate and other communication parameters are set consistently for master and slave. Also reports loss of characters in buffer when hardware storage is exceeded.
36	Bad commutation table. The data points in the back EMF table HV[N] do not form a valid back EMF function.	This error normally occurs while attempting MO=1. At MO=0, the HV[N] table can be updated, so its consistency is not required.
37	Two or more Hall sensors are defined for the same location.	One of the following: CA[4]=CA[5], CA[4]=CA[6] or CA[5]=CA[6].
39	Motion start specified for the past. The time requested for synchronized motion has elapsed.	
41	Command not supported by this product. An attempt has been made to assign an illegal value to the command.	YA[4]=3 attempts to set a type of auxiliary encoder as analog. It causes this error because SimplIQ for Steppers drives do not work with analog encoders.

Error Code	Error String / Description	Example / Remedy
42	No such label. The program does not contain a label with the specified name.	XQ##FOO will return this error if neither the label ##FOO nor the function with the name FOO exists in the user program.
43	CAN state machine fault (object 0x6040 on DS402).	Reset the fault by sending the control word through CAN communication (the value of CAN object 0x6040 must be set to 0x80). Refer to the description of the 0x6040 and 0x6041 objects in the Elmo *CANopen Implementation Manual*.
45	Returned error from subroutine. Occurs when a return op-code has no valid address to return to.	
46	May not use multicapture homing mode with stop event. Occurs when trying to set multiple capture events with a STOP between events.	The following cannot be set: HM[4]=0; HM[1]=3.
47	User program does not exist. XQ or XC returns this error if a program has not been loaded to and successfully verified by the drive.	
50	Stack overflow. A CPU exception was detected. This error reflects either a hardware problem or a faulty power supply.	Use the CD command to determine the details of what has occurred. If “Called handler” is “none” and “Failure address” is non-zero, then TS is too short and there was a real-time overflow. Record the entire string of the CD command and call your service center for technical support.
53	Only for current. Command is applicable only in torque control modes UM=1 or 3.	TC=2 (Set torque to 2A) is an error in UM=2, because in this mode, the torque command is set automatically by the controller so as to achieve the desired speed.
54	Bad database. Cannot start the motor, because the setup data is not consistent.	If CA[4]==CA[5], two physical Hall sensors are defined as the same logical sensor, thereby preventing powering the motor.

Error Code	Error String / Description	Example / Remedy
55	Bad context. A command that is not applicable in the present context has been attempted.	This error is caused by privileged commands used in auto-setup sessions.
56	The product grade does not support this command.	User may have attempted to set or activate features that are available only for the Advanced SimplIQ for Steppers models.
57	Motor must be off. This command cannot be used when the motor is on.	CA[25]=1 sets the order of firing the motor phases, thereby controlling the motor direction. This parameter cannot be set while the motor is on, because it will immediately destabilize the feedback loop.
58	Motor must be on. This command cannot be used when the motor is off.	PA=1000 is an error if MO=0. The absolute position reference is automatically set to the present position at MO=1, so that setting PA at MO=0 is pointless.
60	Bad unit mode. Something not supported in this unit mode has been attempted.	PT=5 is an error in UM=1 because PT motion requires position control.
61	Database reset. The database has been restored to factory defaults after the parameters loaded from the flash memory failed a consistency check.	This error may occur after upgrading the drive version, if the newer version uses a different database structure.
64	Cannot set the index of an active table.	When the ECAM table is active, an index cannot be changed. Only a location beyond the active table limits may be changed.
65	Disabled by SW. Motion could not begin because a switch programmed to abort motion was active when MO=1 was tried.	Check the IL[N] switch definition settings and compare them to the actual switch reading (use the IP command).
66	Drive not ready. The motor could not be powered due to: ▪ Over- or under-voltage ▪ Over-temperature ▪ Short circuit (a shorted motor or a hardware problem) ▪ Hall sensor problem	Check the servo drive status (SR command or MF command)

Error Code	Error String / Description	Example / Remedy
67	Recorder is busy. A recording process is in progress and the recorder settings cannot be changed. Recorded data cannot be fetched.	Let the recorder complete its job, or use the RR=0 command to kill the recording process.
69	Recorder usage error. Something illegal was attempted with the recorder.	RC=2; RR=2 and later BH=1 is an error because an attempt is made to bring a vector that was not recorded.
70	Recorder data invalid. Cannot upload recorded data because the recorder contains no valid data.	Recorder settings (such as RC=n) have been changed since the last records were made or the recorder has not been operated at all since power up.
71	Homing is busy. Cannot change the modulo count (XM or YM) while homing is in progress.	Terminate homing processes using HM[1]=0, HY[1]=0.
72	Must be even.	(XM[2] - XM[1]) = 5 is an error because the difference is an odd number.
73	Please set position. An attempt to set the position counts modulo to a smaller number than the present position was made.	PX=2000; XM[1]=-500, XM[2]=500 is an error because the PX value is out of range.
77	Buffer too large. A string command that is too long (more than 255 characters in a single command) has been sent.	Check the command syntax.
78	Out of program range. An attempt to load a program larger than the drive storage capabilities has been made.	The amount of allocated memory for the user program is stated in the drive *User Manual*.
80	ECAM data inconsistent. The jumps between consecutive ECAM table points are greater than 32, 767 counts and therefore cannot be interpolated.	
81	In “Quick stop” mode. Occurs only if a CAN master used the DS402 standard control word to block motor movements.	In “Quick stop” mode, it is impossible to begin a software motion. Use CAN 0x6040 object to release the “Quick stop” state.
82	Program is running. Cannot load a new program, compile a program or start program execution.	Wait until the program finishes, or use the HP command or KL command to stop the program.

Error Code	Error String / Description	Example / Remedy
83	CMD not for program. An attempt has been made to use a command (such as XQ, DL, LS or DF) that has a NotProgram flag.	The next expression XQ##START; inside a user program is an error because this command has a NotProgram flag.
84	The system is not in point to point mode.	A PR (position relative) cannot be set in non-PTP mode, because it has no reference position from which to start.
90	CAN state machine is not ready (object 0x6041 on DS402).	Set the drive to the "Switched on" state machine by setting the relevant transitions to the control word, object 0x6040. Refer to the description of NMT services in the Elmo *CANopen Implementation manual*.
93	There is a wrong initiation value for this command.	Reset queue length before updating queries.
95	Too large for modulo setting.	The modulo range is inconsistent with the ER[3] value. Refer to the ER[N] command.
96	User program time out. Execution of a single user program line was more than expected (more than 3 seconds). The SimplIQ for Steppers drives stops program execution.	
97	RS232 receive buffer overflow. Characters arrived through RS-232 at too high a rate, causing internal storage to exceed its capacity. No more space is left to store new characters.	
99	The auxiliary feedback entry does not configure as output during the activation of Output Compare	
100	The requested PWM value is not supported	The PWM frequency that was requested cannot be used with the drive.
101	Abortive attempt to read a position value from the absolute position sensor. CD command also indicates failure details.	

Error Code	Error String / Description	Example / Remedy
105	Speed loop KP out of range. Value of KP[2] or one of KG[64]…KG[126] is out of numeric range.	
106	Position loop KP out of range. Value of KP[3] or one of KG[127]…KG[189] is out of numeric range.	
107	Inconsistent data for commutation setup, for example if CA[17] defines commutation by Hall sensors while CA[20] asserts that Hall sensors are absent.	Workout the commutation tuning with the tuning tools that comes with the device. They will yield consistent setup.
110	Bad motion tables. The motion tables are entered via the TV command and synchronized by SI.	Use the tuning software to re-form the tables.
111	KV[N] vector is invalid. Invalid values in KV[N] parameters.	Refer to the Advanced Filter chapter in the *SimpIIQ for Steppers Application Note*. If the vector was configured by the Composer auto-tuning wizard, email Technical Support for assistance.
112	KV[N] defines scheduled block but scheduling is off. Invalid values in KV[N] parameters.	See the syntax of KV[N] parameters in the Advanced Filter chapter of the *SimpIIQ for Steppers Application Note*. If the vector was configured by the Composer auto-tuning wizard, e-mail Technical Support for assistance.
113	Exp task queue is full. Internal error during auto-tuning process.	
114	Exp task queue is empty. Internal error during auto-tuning process.	
115	Exp output queue is full. Internal error during auto-tuning process.	
116	Exp output queue is empty. Internal error during auto-tuning process.	
117	Bad KV setting for sensor filter. Invalid setting for KV[76]…KV[87].	See the KV command section of this manual.
118	Bad KV vector	This can happen when KV parameters are not set according to the correct feedback with either length or value restriction.

Error Code	Error String / Description	Example / Remedy
119	Bad Analog sensor Filter	When the filter KV, set for analog feedback, is beyond its legal range.
120	Bad number of blocks for Analog sensor filter	When the analog sensor filter contains a wrong number of blocks.
121	Analog sensor is not ready	When the initiation procedure of the Analog sensor was not completed, and the motor is being enabled.
123	Motion control processor does not respond. Possibly, firmware has not been well loaded.	
127	Modulo range must be positive. XM[2] is less or equal to XM[1] or YM[2] is less or equal to YM[1].	
128	Bad variable index in database - internal compiler error. Index of the variable in the database is not correct.	An internal compiler error occurs due to a corrupted database. In such a case, email Technical Support for assistance. Attach the Composer date and version (found in the Help menu) and the program you attempted to compile.
129	Variable is not an array. Cannot access a scalar variable defined according to its index in square brackets as an array in the user program.	Assume that a scalar variable has been defined in the user program as a: long a; The expression a[0]=1; is wrong because a is defined as scalar and not an array.
130	Variable name does not exist. For *SimpIIQ for Steppers* internal use.	
131	Cannot record local variables. For *SimpIIQ* drive internal use.	
132	Variable is not an array. For *SimpIIQ* drive internal use.	

Error Code	Error String / Description	Example / Remedy
133	Mismatched number of user/system function input arguments. An attempt was made to call a user/system function with the number of input arguments different from that defined.	• rnd(4.6,7.7) ; This expression is wrong, since system function rnd() expects only one input argument. • Calling user function by XQ command with a number of input arguments different from that defined in the user program.
134	Cannot run local label with XQ command.	XQ##START; when START is defined in the user program inside a user function it is consider to be a local label and therefore it is illegal to use it with the XQ command.
137	Program already compiled. An attempt was made to download a user program before the previous one was erased.	Use the CP command before downloading a new user program.
139	The number of breakpoints exceeds the maximum number.	
140	An attempt to set/clear breakpoint at the non-relevant line. Internal IDE error.	For every line of the text program, there is a corresponding line of compiled code. This error appears during an attempt to set a breakpoint at a non-corresponding line of compiled code.
141	Boot identity parameters section is not clear. Internal error during download of boot identity parameters.	
142	Checksum of data is not correct. Internal error during download of boot identity parameters.	
143	Missing boot identity parameters. Internal error during download of boot identity parameters.	
144	Numeric stack underflow. An attempt has been made to retrieve an entry from an empty stack.	

Error Code	Error String / Description	Example / Remedy
145	Numeric stack overflow. An attempt has been made to push a value to the numeric stack when it is full.	▪ User program contains very complex code requiring more stack space than is available. It may also be that too many subroutines are called. ▪ An expression in the command line of the interpreter is too complex; it calls too many functions, so that the numeric stack has overflowed.
146	Expression stack overflow. An attempt has been made to push a value to the expression stack when it is full.	An expression in the command line of the interpreter is too complex: it calls too many functions, so that the expression stack has overflowed.
147	Executable command within math expression. An attempt has been made to assign an executable command.	BG=3; is wrong because BG is an executable command and cannot be assigned.
148	Nothing in the expression. An attempt has been made to evaluate an empty expression.	AC=; is wrong because the assigned value is missing.
149	Unexpected sentence termination. An expression terminator appears in the middle of the expression.	5+3+;
150	Sentence terminator not found. The expression is too long to be evaluated (exceeding the maximum length).	Try to shorten the expression.
151	Parentheses mismatch. There is a mismatch between opening and closing parentheses. Pertains to both parentheses and brackets.	sin(2; is wrong because a closing parenthesis is absent.
152	Bad operand type. There is a mismatch between the actual value type and the expected value type.	▪ The DB command syntax is wrong (this command requires strict syntax; trying to set an unexpected floating point value causes this error). ▪ An internal compiler error has occurred due to a mismatch between operand type and its addressing mode. Contact Technical Support.

Error Code	Error String / Description	Example / Remedy
154	Address is out of data memory segment. Variable address in the data segment exceeds the data segment size.	This is an internal compiler error caused by compiled code that has become corrupted. In such a case, email Technical Support for assistance. Attach the Composer date and version (in the Help menu) and the program you attempted to compile.
155	Beyond stack range. Compiled code contains a pointer to the stack entry, exceeding the actual stack range (STACK_IMMEDIATELY addressing method).	This internal compiler error is caused by corrupted compiled code. In such a case, email Technical Support for assistance. Attach the Composer date and version (in the Help menu) and the program you tried to compile.
156	Bad opcode. Compiled code contains mismatched addressing mode.	Internal compiler error caused by corrupted compiled code. In this case, email Technical Support for assistance. Attach the Composer date and version (in Help menu) and the program you tried to compile.
157	No available program stack. An attempt was made to run too many user programs simultaneously.	For future use.
158	Out of flash memory range. Failure in download and upload process: an attempt to access flash memory because of its size.	Try to use IDE tools for down-loading or uploading.
159	Flash verify error. Failure in download and upload process: checksum does not match.	Possible hardware problem. Contact Technical Support.
160	Program aborted by another threat. Failure while running one virtual machine aborts all other virtual machines.	For future use.
161	Program is not halted. Execution of a command that requires user program to be halted.	Activation of XC command while the virtual machine is not in a halted state.
162	Badly formatted number. Floating point number exceeds the valid range supported by SimplIQ for Steppers.	

Error Code	Error String / Description	Example / Remedy
164	EC command (not an error).	For internal use.
165	An attempt to access serial flash while busy. Failure on reading serial flash memory, possibly due to a hardware problem.	Contact Technical Support.
166	Out of modulo range.	XM[1]=-1000, XM[2]=1000 and PA=2000 is an error because the modulo range is [-1000…999]. Therefore, the position of PA=2000 can never be reached.
167	Infinite loop in for loop - zero step	k=1:0:10; causes this error.
168	Speed too large to start motor. MO=1 or motor started with Enable switch while motor was rotating too quickly.	Starting a running motor may fail if the back EMF is too high and induces an immediate excessive motor current.

Table 3-10: **Processing Error Codes**

Attributes:

Type:	Status report, Integer
Source:	RS-232, CANopen
Restrictions:	None
Unit modes:	All
SimplIQ:	Modified

See also:
MF, SR

EF[N] - Encoder Filter Frequency

Purpose:
Filters encoder signal in order to improve its noise immunity. Because the logic of the quadrature decoder must sense transitions, the inputs are first run through a glitch filter. This filter has a digital delay line that samples one, three, or six time points on the signal and verifies that all of the samples are at a new state before outputting the new state to the internal logic. The sample rate of this delay line is programmable, to adapt to a variety of signal bandwidths.

When an analog encoder is used, the basic signal, before interpolation, is filtered using the same method.

EF[N] sets the sample rate of the corresponding digital glitch filter: EF[1] for the main encoder and EF[2] for the auxiliary encoder. If EF[N] is zero the digital filter is bypassed.

If EF[N] is large, the encoder reading noise immunity will be better, but true fast transitions (occurring by fast speed) may be dismissed as false. A number that is too small may cause the counting of noise pulses.

The formula for calculating EF[N] is $EF[N] = \frac{10^7}{\text{max expected speed in counts/second}}$.

Example:
Suppose that the maximum speed of a motor is 1000 rpm and that the motor is equipped with an encoder with 1000 lines (4000 counts/rev with resolution multiplication). The expected speed is

$$\frac{4000 count/rev \times 1000 rpm}{60 \text{sec/min}} = 6.67 \times 10^5 count/\text{sec}$$ and we set $$EF[1] = \frac{10^7}{667500} \approx 15$$

Notes:

- When an interpolator is used, the time difference between consecutive signal changes may sometimes be much shorter. This is dictated by the interpolator's specifications and not by the motor speed.
- When working with an analog encoder, always set EF[1]=18.

Attributes:		
	Type:	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	0 (RS), Non-volatile
	Range:	[0...127]
	Index range:	[1, 2]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Modified

EM[N] - ECAM Parameters

Purpose:
Defines the behavior of ECAM (Electronic CAM) motions. With ECAM, the position reference to the drive is made an arbitrary function of the external encoder reading. The ECAM parameters apply only to position modes (UM=3 and 5) and when the position reference is derived from the auxiliary encoder input (RM=1, FR[3]=non-zero).

Parameter	Description
EM[1]	ECAM mode 0: Inactive 1: Linear ECAM 2: Cyclical ECAM Set EM[1] to enter a change in the EM[2], EM[3], EM[4], EM[5] and EM[7] parameters.
EM[2]	Last valid index of the ECAM table. Maximum value is 1024.
EM[3]	Starting position: the value of the input to the ECAM function for which the output of the ECAM function will be ET[EM[5]] (ET of EM[5]).
EM[4]	Auxiliary input (ΔPY) distance between consecutive points in the ECAM table ET[N].
EM[5]	First valid index of the ECAM table.
EM[6]	Index for the next head pointer when using CAN for fast ECAM table loading.
EM[7]	Last segment shortening. Used to generate an ECAM table with an input range that is not an integer multiple of EM[4].
EM[8]	Read-only report of position in the ECAM table. When ECAM motion is not active, EM[8] reports 0.

Table 3-11: **ECAM Parameters**

Notes:

- When EM[1]=1 or EM[1]=2, the active ECAM table entries — ET[EM[5]] . . . ET[EM[2]] — cannot be changed. Other members of the ET[N] array *may* be changed.
- Parameter EM[6] takes effect immediately.
- Parameters EM[2], EM[3], EM[4], EM[5] and EM[7] are activated only when EM[1] is set. In this manner, the next work segment can be programmed while the present work segment is executing.

- Setting EM[1] to 1 or 2 will fail if EM[2] is less than or equal to EM[5], or if EM[4] is less than or equal to EM[7].
- Changing the last segment with EM[7] may cause a reference jump.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	Advanced model only
	Default values:	EM[1]=0, EM[2]=2, EM[3]=0, EM[4]=1000, EM[5]=1, EM[6]=1, EM[7]=0, EM[8]=0 (RS), Non-volatile
	Range:	EM[1]: [0…2] EM[2]: [2…1024] EM[3]: Position counter range EM[4]: [1…32,000] EM[5]: [1…1023] EM[6]: [1…1024] EM[7]: 1…32,000]
	Index range:	Write: [1…7] Read: [1…8]
	Unit modes:	UM=3, 4, 5
	Activation:	See previous Notes
	SimplIQ:	Similar

See also:
RM, FR[N]

Application note:
The Position External Reference Generator

EO - Echo Mode

Purpose:
Sets or resets the communication echo mode, which is used for communication checks.

- EO=1 enables echo mode
- EO=0 disables it.

With RS-232 communication, the EO command sends an immediate echo character for every terminal character.

Attributes:		
	Type:	Parameter, Integer
	Source:	RS-232
	Restrictions:	None
	Default value:	1, Volatile
	Range:	[0, 1]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

ER[N] - Maximum Tracking Error

The Tracking Error is the difference between the command and its feedback.

Purpose:

- ER[2] defines the maximum allowed velocity error (abs(DV[2]-VX)) in counts/second. If the error exceeds this value, the motor is automatically disabled and the Error Limit fault is activated. ER[2] affects speed control only. ER[2] is inactive for position feedback modes (UM=4, UM=5).
- ER[3] defines the maximum allowed position error in counts:
  for UM=5: abs(DV[3]-PX)
  for UM=4: abs(DV[3]-PY)
  If the error exceeds this value, the motor is automatically disabled and the Error Limit fault is activated.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	ER[2]: None ER[3] - Refer to note below
	Default values:	ER[2]=400,000, ER[3]=400,000 (RS), Non-volatile
	Range:	[0…20,000,000]
	Index range:	[2, 3]
	Unit modes:	UM=2 for ER[2], UM=4, 5 for ER[3]
	Activation:	Immediate
	SimplIQ:	Similar

The ER[3] value is restricted to modulo settings. The values ranges are:
For UM=5: [0..(XM[2]-XM[1])/4-1]
For UM=4: [0..(YM[2]-YM[1])/4-1]
In case of failure, an error is set during the next motor enabling (MO=1).

Typical applications:
Decrease ER[N] as much as possible in order to use it as a protection mechanism in case of control failure, such as when the feedback signal is lost.

See also:
MF, MO, SR

Application notes:

ER[2]: The Speed Controller

ER[3]: The Position Controller

ET[N] - Entries for ECAM Table

Purpose:
In the ECAM process, the position reference is set to a tabulated function, called the ECAM function, of the external inputs. The ET[N] vector stores the tabulated values of the ECAM function.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None (see Notes below)
	Default values:	0 (RS), Non-volatile
	Range:	$[(-2^{30}+1)\ldots(2^{30}-1)]$
	Index range:	[1…1024]
	Unit modes:	UM=3, 4, 5
	Activation:	Immediate
	SimplIQ:	Similar

Notes:

- When the motor is enabled (MO=1) and the ECAM table is running (EM[1]=1), you may set entries that are not in an active part of the table. This enables on-the-fly programming of the next motion. Refer to the EM[N] command for active table descriptions.
- With CAN communication, you may program the ECAM table with a fast Auto-increment mode.
- When the ECAM table is not used, ET[N] can be used as a general-purpose non-volatile memory.

See also:
EM[N], UM, RM

Application note:
The External Position Reference Generator

FF[N] - Feed Forward

Purpose:
Defines the feed forward control effort.

FF[1] defines the ratio between the acceleration reference and the direct current command.

FF[2] defines the ratio between the speed reference and direct commanding of the speed controller.

Attributes:	**Type:**	Parameter, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	FF[1]=0 (RS), Non-volatile
		FF[2]=1 (RS), Non-volatile
	Range:	[0.0…32000.00]
	Index range:	[1, 2]
	Unit modes:	UM=4, 5
	Activation:	Immediate
	SimplIQ:	Slightly modified

Notes:

- For UM=3 (open loop stepping), the HT[2] command defines the ratio between acceleration and drive.
- For most UM=5 applications, FF[2]=1.
- For most UM=4 applications, FF[2] is the number of counts traveled by the main (speed) feedback, while the position (auxiliary) feedback travels one count.

Examples:
Suppose that a gear motor with a reduction ratio of 5 drives per load. The motor has an encoder with 1000 lines. The motor speed is used for the inner feedback loop. The load position, measured by an encoder with 2000 lines, is used as feedback for the outer loop. To prevent a steady-state error at constant speed,
set: $FF[2] = \frac{1000 * 5}{2000} = 2.5$.

See also:
UM

Application notes:

Unit Modes

The Position Controller

The Speed Controller

FR[N] - Follower Ratio

Purpose:

- FR[1] defines the follower ratio for current (UM=1).
- FR[2] defines the follower ratio for velocity (UM=2).
- FR[3] defines the follower ratio for position (UM=3, 5).

When UM=1, the auxiliary reference is composed of the analog input and external PWM signals. The FR[1] parameter scales the ratio between the Duty Cycle of the PWM signal and the reference to the current loop (UM=1, RM=1). FR[1] may be changed on-the-fly at any time.

When UM=2, the auxiliary reference is composed of the analog input and the auxiliary feedback readout or external PWM signal.

FR[2] can be used as a follower ratio of the master motor's quadrature encoder or as a follower ratio of the PWM Duty.

When UM=3 or UM=5, the auxiliary reference is derived from the auxiliary feedback readout. The FR[3] parameter scales the ratio between the auxiliary feedback position and the reference to the position loop (UM=3, 4, 5, RM=1, EM[1]=0), or the input to the ECAM table (UM=3, 4, 5, RM=1, EM[1]=1, 2).

FR[3] may be changed on-the-fly at any time. For EM[1]=0 (follower) and EM[1]=2 (cyclical ECAM), changing FR[3] does not imply an abrupt change to the external position controller reference. For EM[1]=1 (linear ECAM), changing FR[3] *does* imply an abrupt change in the external position controller reference.

FR[3] can be modified without a motion jump when the software reference is not active (MS<1). Then the software position reference is corrected to avoid a possible motor jump.

Attributes:		
	Type:	Parameter, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0: none MO=1: RM=1 and MS<2
	Default value:	1 (RS), Non-volatile
	Range:	[-32,000…32,000]
	Index range:	[1, 2, 3]
	Unit modes:	UM=1, 2, 3, 5
	Activation:	Immediate

See also:
PY, RM, YM[N], YA[N], PW[N]

Application notes:

The Position Reference Generator

The External Position Reference Generator

GS[N] - Gain Scheduling

Purpose:
Defines the gain scheduling process.

SimpIIQ drives contain several sets of controller parameters. The ability to select the best control parameters on-the-fly is called gain scheduling.

The following table lists the gain scheduling parameters. Unused indices are reserved for compatibility with older drives.

Parameter	Description	Values
GS[1]	Minimum speed command for speed and dual gain scheduling (counts/sec)	0…16*62*256 internal speed units (1 speed unit = counts/Ts/2^16)
GS[2]	Use scheduled gains: 0: No 1..16: Manual 64: Automatic	0..16 64
GS[4]	Upward gain of gain scheduling filter	0…32,767
GS[5]	Downward gain of gain scheduling filter	0…32,767
GS[9]	NL factor for position controller: The maximum deceleration that the motor can apply. Used to assure that the position controller commands only speeds that can be stopped on time without overshoot.	0…500000000
GS[10]	Position error coefficient for position gain scheduling to raise gains	0…1,200
GS[15]	Gain scheduling step [in speed units]: 0: 16384 1: 8192 N: $16384/2^N$	0…10

Table 3-12: **Gain Scheduling Parameters**


Under most circumstances, this command is used only by the tuning environment.

Note:

- The GS[N] array for SimplIQ for Steppers includes 16 gain scheduling controllers. In SimplIQ, it included 64 controllers.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0
	Default values:	
		GS[4]=30,000, Non-volatile
		GS[5]=30,000, Non-volatile
		GS[9]=60,000,000, Non-volatile
		Other defaults=0, Non-volatile
	Range:	According to previous table
	Unit modes:	UM=2, 4, 5
	Activation:	Immediate
	SimplIQ:	Modified – see notes

See also:

KG[N], KP[N], KI[N]

Application Notes:

Gain scheduling

GS[9]: Position controller parameters

GS[10]: The position controller

HL[N] - Over-speed Limit and Position Range Limit

Refer to LL[N] - Low Feedback Limit.

HM[N] - Homing, Capture and Flag

Purpose:
Sets the parameters of the main homing and capture process, by which the drive sets a trap for a user-defined event. When the event occurs, the *SimpIIQ* can:

- Modify a position counter (homing)
- Log the exact position of the event (capture)
- Flag a digital output (flag)

An event is a change in a digital input signal. The polarity of the change is defined by the IL[N] command. Values in HM[3] are duplicated for compatibility reasons.

HM[N] (Index)	Value	Description
1 Activation mode	0	Disarm homing process. HM[1] is automatically reset to 0 when homing is complete.
	1	Arm homing process. The sequence is activated according to the last HM[2] to HM[6] values. HM[7] and HM[8] are cleared.
2 Absolute value		Value to load, according to method of HM[5]. Absolute value is limited to position counter range.
3 Event definition	0	Immediate: Trigger is the receipt of HM[1]=1.
	1/2	Event according to Main Home switch (capture).
	3	High transition[1] of Index pulse (capture).
	4	Low transition[2] of Index pulse (capture). **Note:** This option must ***never*** be used for analog coders.
	5/6	Event according to defined FLS switch.
	7/8	Event according to defined RLS switch.
	9/10	Event according to defined DIN1 switch.
	11/12	Event according to defined DIN2 switch.
	13/14	Event according to defined DIN3 switch.
	15/16	Event according to defined DIN4 switch.
	17/18	Event according to defined DIN5 switch.
	19/20	Event according to defined DIN6 switch.
	21/22	Event according to defined DIN7 switch.
	23/24	Event according to defined DIN8 switch.
	25/26	Event according to defined DIN9 switch.
	27/28	Event according to defined DIN10 switch.

[1] Index input level changes from low to high.
[2] Index input level changes from high to low.

HM[N] (Index)	Value	Description
4 After-event behavior. Defined as the time in which HM[1] decreases to 0.	0	In UM=2, 3, 4, 5: Stop immediately using the SD deceleration value. In torque mode (UM=1), do nothing.
	1	Set digital output, equivalent to OP=HM[6].
	2	Do nothing.
5 What to set for PX during event	0	Absolute setting of position counter: PX=HM[2].
	1	Relative setting of position counter: PX=PX (at event) -HM[2]
	2	Do nothing.
6 Output value		Digital output value if HM[4]=1. Only outputs defined as general output are affected.
7 PX captured value		The captured value of PX (read only). The position value is captured before PX is changed according to HM[5].
8 PY captured value		The captured value of PY (read only). The position value is captured at the next controller sampling time and therefore may be inaccurate to ($4*VX*TS*10^{-6}$) counts.

Table 3-13: **HM[N] Command Values**

Notes:

- SimplIQ for Steppers drives have a different number of digital inputs. The value of HM[3] may differ according to that. Please consult the drive's *Installation Guide* for details.
- HM[2] - HM[6] can be changed during the home search procedure. The activation of the parameters is considered upon reception of the next HM[1]=1 (or higher).
- If HM[2] is set to a value beyond XM[1] and XM[2], the actual main position will not be updated when homing is complete.
- Each homing event is attached to a predefined functionality (FLS, General Purpose, Home and so on). If the corresponding input is not defined first, the homing procedure may never end. Refer to the IL[N] command.
- The homing and capture procedures can be carried out in any unit mode (UM=1, 2, 3, 4, 5) and reference mode (RM=0, 1).
- In external reference mode (RM=1), when HM[4]=0, the software portion of the reference is stopped, while the external portion is not. In such cases, the motor continues to move according to the analog reference.

- Homing can be safely carried out in PTP and jog position motion modes. With PT and PVT modes, the online reloading of the position counter can lead to an immediate, automatic MO=0 due to an excessive position error.
- When the Index or Home signal captures PX, the PY captured value is taken at the next position controller sampling time (4 TS period). It may differ from the PY value at the capture time by up to $4*TS*10^{-6}*VY$ counts.
- When the Index or Home signal captures PX, the digital output of HM[6] is set only at the next position controller sampling time (4 TS period).
- Negative index polarity should ***never*** be used with analog encoders. ***SimplIQ*** will not detect an error but the results may be incorrect.

Attributes:		
	Type:	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	0, Volatile
	Range:	HM[1]: [0, 1] HM[2]: According to PX command HM[3]: [0…24] (see the first note, above) HM[4]: [0…2] HM[5]: [0…2] HM[6]: According to OP command HM[7 - 8]:Read only, according to PX, PY
	Index range:	[1…8]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ for Steppers:	Similar

See also:
HY[N], XM[N], IL[N], OL[N], PX, PY, EF[N], IF[N], SD

Application Notes:

Homing with analog index

HP - Halt Program Execution

Purpose:
Stops the execution of the user program and the automatic routines. The HP command freezes the status of the program and does not reset it. A later XC command will resume the program. Pending interrupts will remain pending.

An HP command issued when no program is running does nothing and sets no error code.

Attributes:	**Type:**	Command, No value
	Source:	RS-232, CANopen
	Restrictions:	None
	Unit modes:	All
	Activation:	Immediate
	SimplIQ for Steppers:	Similar

The HP command, together with a later XC command, may be used if a communicated command sequence must be executed consecutively, without program interference.

See also:
KL, XQ, XC

SimplIQ Programming and Language Guide

HT[N] – Holding torque

Purpose:

This command applies to the motor current in open loop stepping mode (UM=3)

- HT[1] defines the holding torque, in Amperes.
- HT[2] defines the torque addition due to speed, Amp/(Count/Sec)
- HT[3] defines the torque addition due to acceleration, Amp/(Count/Sec^2)


HT[2] and HT[3] are defined differently for open loop and closed loop stepper control (see PN[9]).

- For open loop mode, HT[2] and HT[3] are defined for PN[4] counts/electrical rev. or PN[4]xS/4 counts/mechanical rev., where S is the number of steps.
- For closed loop mode, HT[2] and HT[3] are defined for the encoder counts.

Attributes:		
	Type:	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default value:	0 (RS), Non-volatile
	Range:	HT[1]: 0..CL[1], HT[2]>=0, HT[3]>=0
	Index range:	[1, 2, 3]
	Unit modes:	UM=3
	Activation:	Immediate
	SimplIQ:	New

See also:
PN[N], PF[N], UM

Application note:
Torque and position reference generation for open loop stepper

HX - Hexadecimal Mode

Purpose:
Sets or resets the hexadecimal mode for reporting integer parameter values.

- With HX=0, integers are reported as decimal numbers.
- With HX=1, integers are reported as hexadecimal numbers.

HX does not affect floating-point reports.

The HX parameter allows easy reading of the digital inputs (IP), servo drive status (SR), motor faults (MF), recorder settings (RC) and other variables that have the bit-field attribute.

The HX parameter is not required for setting values. The commands BH=1024 and BH=0x400 are equivalent, as 0x400 equals its decimal equivalent of 1024.

Attributes:		
	Type:	Parameter, Integer
	Source:	RS-232
	Restrictions:	None
	Default value:	0, Volatile
	Range:	[0, 1]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

HY[N] - Auxiliary Homing, Capture and Flag

Purpose:
Sets the parameters of the auxiliary homing and capture process, by which *SimpIIQ* sets a trap for a user-defined event. When the event occurs, the drive can:

- Modify the auxiliary position counter (homing)
- Log the exact position of the event (capture)
- Flag a digital output (flag)

An event is a change in a digital input signal. The polarity of the change is defined by the IL[N] command. Values in HY[3] are duplicated for compatibility reasons.

HY[N] (Index)	Value	Description
1 Activation mode	0	Disarm homing process. HY[1] is automatically reset to 0 when homing is complete.
	1	Arm homing process. The sequence is activated according to the last HY[2] to HY 6] values. HY[7] and HY[8] are cleared.
2 Absolute value		Value to load, according to method of HY[5]. Absolute value is limited to position counter range.
3 Event definition	0	Immediate: Trigger is the receipt of HY[1]=1.
	1/2	Event according to Auxiliary Home switch (capture).
	3	High transition of Index pulse (capture).
	4	Low transition of Index pulse (capture).
	5/6	Event according to defined FLS switch.
	7/8	Event according to defined RLS switch.
	9/10	Event according to defined DIN1 switch.
	11/12	Event according to defined DIN2 switch.
	13/14	Event according to defined DIN3 switch.
	15/16	Event according to defined DIN4 switch.
	17/18	Event according to defined DIN5 switch.
	19/20	Event according to defined DIN6 switch.
	21/22	Event according to defined DIN7switch.
	23/24	Event according to defined DIN8 switch.
	25/26	Event according to defined DIN9 switch.
	27/28	Event according to defined DIN10 switch.
4 After-event behavior.	0	In UM=2, 3, 4, 5: stop immediately, using SD deceleration value. In torque mode (UM=1), do nothing.
	1	Set digital output, equivalent to OP=HY[6].

HY[N] (Index)	Value	Description
Defined as the time in which HY[1] decreases to 0.	2	Do nothing.
5 What to set for PY during event	0	Absolute setting of position counter: PY=HY[2].
	1	Relative setting of position counter: PY = PY (at event) -HY[2]
	2	Do nothing.
6 Output value		Digital output value if HY[4]=1. Only outputs defined as general output are affected.
7 PY captured value		Captured value of PY, before any modification by the HY[N] command (read only).
8 PX captured value		Captured value of PX (read only).

Table 3-14: **HY[N] Command Values**

Notes:

- HY[2] - HY[6] can be changed during the home search procedure. The activation of the parameters is considered upon reception of the next HY[1]=1 (or higher).
- If HY[2] is set to a value beyond YM[1] and YM[2], the actual auxiliary position will not be updated when homing is complete.
- Each homing event is attached to a predefined functionality (FLS, General Purpose, Home and so on). If the corresponding input is not defined first, the homing procedure may never end. Refer to the IL[N] command.
- The homing and capture procedures can be carried out in any unit mode (UM=1, 2, 3, 4, 5) and reference mode (RM=0, 1).
- In external reference mode (RM=1), when HY[4]=0, the software portion of the reference is stopped, while the external portion is not. In such cases, the motor continues to move according to the analog reference.
- Homing can be safely carried out in the PTP and jog position motion modes. With PT and PVT modes, the online reloading of the position counter can lead to an immediate, automatic MO=0 due to an excessive position error.
- When the Index or Home signal captures PY, the captured PX value is taken at the next position controller sampling time (4 TS period). It may differ from the PX value at the capture time by up to $4*TS*10^{-6}*VY$ counts.
- When the Index or Home signal captures PY, the digital output of HY[6] is only set at the next position controller sampling time (4 TS period).

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	0, Volatile
	Range:	HY[1]: [0, 1] HY[2]: According to set PY command HY[3]: [0…24] HY[4]: [0…2] HY[5]: [0…2] HY[6]: According to OP command HY[7 - 8]: Read only according to PX, PY
	Index range:	[1…8]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

See also:
HX, XM[N], IL[N], OL[N], PX, PY, EF[N], IF[N]

IB[N] - Input Bits Array

Purpose:
Provides read access to digital input bits.

IB[N] reports the status of the corresponding input bits, according to the definition in IP. If IB[N] is "1", the corresponding Nth bit in IP is logically active.

Use the IB[N] command to reference general purpose inputs, limit switches and other indications (such as Stop, Begin or Enable) individually.

IB[N] may be more convenient than IP for program decisions and branching. However, it is not appropriate for the synchronized reading of several input bits. If a synchronized reading of several digital inputs is desired, use the IP command.

Refer to the IP command for more details about the role of each bit.

Attributes:		
	Type:	Status report, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Index range:	[0…31]
	Unit modes:	All
	SimplIQ:	Similar

IB[N] reports according to the polarity programmed for the relevant digital input by the IL[N] command.

See also:
IP, IL[N]

ID, IQ - Read Active Current and Reactive Current

Purpose:
Gets the active and the reactive components of the motor current, in Amperes.

A brushless motor carries alternating currents in its phases. The alternating currents in the motor phases create a rotating magnetic field, which can be projected in two directions. The first magnetic field component is aligned with the magnetic direction of the rotor; it produces no mechanical torque. The other magnetic field component is perpendicular to the magnetic direction of the rotor and produces all the mechanical torque.

IQ[Amp] is the component of the motor phase currents that creates effective torque. The current controller attempts to make IQ equal to the current command. ID is the component of the motor phase current that does *not* create torque. The current controller tries to null ID.


When the motor is off (MO=0), IQ and ID are not calculated and return zero.

Attributes:	**Type:**	Status report, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Unit modes:	All
	SimplIQ:	Modified

See also:
AN[N], MC, PL[N], CL[N]

Modification from SimplIQ:

Winding shape compensations change the method of calculating ID, IQ

Application notes\:
The input transformation

IF[N] - Digital Input Filter

Purpose:
Filters the drive digital inputs in order to overcome switch bouncing. IF[N] defines a time period in milliseconds. Input pulses of shorter duration than IF[N] are rejected. Pulses longer than IF[N] in milliseconds are sensed.

Each index entry [1 - 10] refers to a digital input [1 - 10] respectively. The input filtering is accomplished by the software. For this reason, in order to ensure that an input pulse is sensed, its length must be IF[N]+WS[4], where WS[4] is the SimplIQ for Steppers sampling time.

Example:
If the SimplIQ for Steppers scan period is 360 microseconds and IF[1]=1, the minimum stable time
for an input change for a sure capture is 1 msec + 360 usec = 1.36 msec.

Attributes:	**Type:**	Parameter, Float
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	0 (RS), Non-volatile
	Range:	[0…1000] msec
	Index range:	[1…10] … refer to the first note
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

Notes:

- SimplIQ drives supports a different number of digital inputs. Accordingly, the index range may differ between drives. Please consult the drive's *Installation Guide* for information about its digital inputs.
- If input 5 or 6 is used as Home input, the corresponding parameter — IF[5] or IF[6]— does not apply. Parameter EF[1] or EF[2] is used instead.
- The IF[N] commands can also be applied in simulation mode – refer to the IL[N] command.

IL[N] - Input Logic

Purpose:
Defines the logic level and functional behavior of the digital inputs. The drive has several non-committed digital inputs. Each of these inputs can be programmed to a specific function and logic level. In addition, the IL[N] function enables the simulation of a digital input. This option is convenient for testing and debugging user programs.

A digital input serves only one dedicated function, which can be reflected in the following commands/features:

- User program auto routine (#@AUTO_##)
- Homing procedure (HM[N], HY[N])
- IP command
- IB[N] command

Notes:

- The response to a digital input is made only according to the definition of IL[N]. For example, if digital input #2 is defined by IL[2] as RLS (Reverse Limit switch), changes in the connector pin of digital input #1 will not be reflected in IB[1] commands. IB[1] will read zero continuously. In this case, IB[11] will reflect this input status.
- When a digital input is activated, the relevant bit in IP/IB[N] is set. Refer to the IP and IB[N] commands for more details.
- Inputs 5 and 6 also serve as high-speed home/capture inputs, used independently by the HM[N] and HY[N] commands. The logic level for the home inputs is also defined by the IL[N] command.
- If inputs 5 or 6 are used as home inputs, the corresponding parameters IF[5] and IF[6] do not apply. The EF[1] and EF[2] parameters are used instead.

The following table summarizes the functions that may be attached to each digital input pin. The function details are given in the next table. The prefix "Hard" indicates that the function applies to the stop manager, not to the motion reference generator. The term "Soft" indicates that the function applies to the motion reference generator. The term "Hard and Soft" indicates that the function applies both to the stop manager and to the position reference generator.

IL[N] Bits	Meaning	Values
0	Logic levels	0: Low level active. The function attached to this switch is activated when *no* current flows through the input opto-coupler.
		1: High level active. The function attached to this switch is activated when current flows through the input opto-coupler.

IL[N] Bits	Meaning	Values
1 - 4	Function behaviors (next table)	0: Inhibit (INH); shut servo driver, freewheel. For RM=1 and UM=1, the *SimpIIQ* drive will retry starting the motor automatically when the inhibit function is released.
		1: Stop immediately under control; hard stop only.
		2: Ignore.
		3: General purpose.
		4: Hard-enable forward direction only (RLS).
		5: Hard-enable reverse direction only (FLS).
		6: Begin.
		7: Stop immediately under control; soft stop only.
		8: Home switch for IL[5] only.
		9: Auxiliary Home switch for IL[6] only.
		10: Simultaneous activation of the hard and soft stop functions (functions 1 and 7).
		11 - Abort.
		12 - 15: Reserved.
5	Simulation mode	0: Read value from digital input pin. 1: Read value from bit 6, regardless of pin state.
6	Simulation value	Value to set for pin if bit 5 is on.
7 - 15	Reserved	

Table 3-15: **IL[N] Functions**

Command Value	Active Level	When Active . . .
IL[N]=0	Low	Shut servo drive, freewheel.
IL[N]=1	High	Shut servo drive, freewheel **Note:** It is high recommended *not* to use this state. The motor may spin when the input wire is cut or disconnected.
IL[N]=2	Low	Stop immediately under control: soft and auxiliary stop.
IL[N]=3	High	Stop immediately under control: soft and auxiliary stop.
IL[N]=4	Low	No function is attached. Ignore the switch.
IL[N]=5	High	No function is attached. Ignore the switch.

Command Value	Active Level	When Active . . .
IL[N]=6	Low	General purpose.
IL[N]=7	High	General purpose.
IL[N]=8	Low	Hard-enable forward direction only (RLS).
IL[N]=9	High	Hard-enable forward direction only (RLS).
IL[N]=10	Low	Hard-enable reverse direction only (FLS).
IL[N]=11	High	Hard-enable reverse direction only (FLS).
IL[N]=12	Low	Begin: activates BG command.
IL[N]=13	High	Begin: activates BG command.
IL[N]=14	Low	Stop immediately under control: soft stop only. Activates the ST command.
IL[N]=15	High	Stop immediately under control: soft stop only. Activates the ST command.
IL[5]=16	Low	Enable the Main Home sequence.
IL[5]=17	High	Enable the Main Home sequence.
IL[6]=18	Low	Enable the Auxiliary Home sequence.
IL[6]=19	High	Enable the Auxiliary Home sequence.
IL[N]=20	Low	Stop immediately under control: stop both software trajectory and auxiliary reference.
IL[N]=21	High	Stop immediately under control: stop both software trajectory and auxiliary reference.
IL[N]=22	Low	Abort motion. Shut servo drive, freewheel.
IL[N]=23	High	Abort motion. Shut servo drive, freewheel.

Table 3-16: **Possible Values for IL[N]**

Function 0: Inhibit (freewheel)
Servo is off (MO=0). The motor is *not* under control. No current is applied through the motor phases. If the motor was previously running, it will continue to coast on its own inertia. The motor fault code (see the MF command) is 0x10. If the unit mode is UM=1 (torque control) or UM=2 (velocity control) and an external command is active (RM=1), a motor restart will be attempted when the switch is "not active." This attempt is made within a few (no less than 10) milliseconds. In addition, when restarting the motor the #@AUTO_ENA automatic routine can be activated.

Function 1: Hard stop immediately under control
The function behavior depends on the unit mode:

UM	Action
Torque (UM=1)	Set torque command to zero.
Speed (UM=2)	Set speed command to zero immediately at the deceleration of the SD parameter.
Position (UM=3, 4, 5)	Slow down to complete stop using the deceleration of the SD parameter.

Table 3-17: **UM Values for Hard Stop**

Function 2: Input is ignored
This serves no function in the system and always reads zero in the IP/IB[N] indications.

Function 3: General purpose (GPI)
No special function. Serves as an uncommitted input. The input may be used in the user program and homing sequences as simple digital input. In addition, general purpose inputs can activate ##AUTO_DIN automatic routines in the user program.

Function 4: Hard-reverse limit switch
The function activates the ##AUTO_RLS routine in the user program. In addition, it has the following unit mode dependent actions:

UM	Action
Torque (UM=1)	Allow only positive torque commands. Negative torque demands yield zero motor current.
Speed (UM=2)	Allow only positive speed command (external or internal). If, at the time of switch sensing, the speed command was negative, the speed command will converge to zero using the stop deceleration (SD).
Position (UM=3, 4, 5)	Allow only positive position command increments (external and internal). If, at the time of switch sensing, the speed was negative, the position command will decelerate to a complete stop using the deceleration of the SD parameter.

Table 3-18: **UM Values for Hard ReverseFunction 5: Hard-forward limit switch**

The function activates the ##AUTO_FLS routine in the user program. In addition, it has the following unit mode dependent actions.

UM	Action
Torque (UM=1)	Allow only negative torque commands. Positive torque demands yield zero motor current.

UM	Action
Speed (UM=2)	Allow only negative speed command (internal or external). If, at the time of switch sensing, the total speed command was positive, the speed command will converge to zero using the stop deceleration (SD).
Position (UM=3, 4, 5)	Allow only negative position command increments (external and internal). If, at the time of switch sensing, the speed was positive, the position command will decelerate to a complete stop using the deceleration of the SD parameter.

Table 3-19: **UM Values for Hard Forward**

Function 6: Begin
The function behaves like a software BG command, activating the ##AUTO_BG routine in the user program. In addition, it has the following unit mode dependent actions:

UM	Action
Torque (UM=1)	Nothing.
Speed (UM=2)	Set software speed command to JV.
Position (UM=3, 4, 5)	Set software position command to the activated motion mode (PA, JV, PT, PV).

Table 3-20: **UM Values for Begin**

Function 7: Software Stop
The function behaves like a software ST command, activating the ##AUTO_ST routine in the user program. In addition, it has the following unit mode dependent actions:

UM	Action
Torque (UM=1)	Nothing.
Speed (UM=2)	Reduce the software speed command to zero, using the deceleration SD.
Position (UM=3, 4, 5)	Set software position command to complete stop, using the deceleration SD.

Table 3-21: **UM Values for Software Stop**

Function 8: Main Home switch
This function activates the ##AUTO_HM routine in the user program. When the function is selected, digital input connector pin #5 serves as the Home/Capture switch for the feedback defined as main. Only IL[5] can be programmed to this function. Refer to the HM[N] command for more information.

Function 9: Auxiliary Home switch
This function activates the ##AUTO_HY routine in the user program. When the function is selected, digital input connector pin #6 serves as the Home/Capture switch for the feedback defined as auxiliary. Only IL[6] can be programmed to this function. Refer to the HY[N] command for more information.

Function 10: Hard and Soft stop
This function activates the ##AUTO_ST routine in the user program. It stops the motor under control, stopping the response to external reference and applying the software ST command simultaneously. This function actually activates function 1 and function 7 simultaneously.

UM	Action
Torque (UM=1)	Set the torque command to zero.
Speed (UM=2)	Reduce the software speed command to zero, using the stop deceleration SD. Reduce the controller speed command to zero, using the deceleration SD.
Position (UM=3, 4, 5)	Set the software position command to a complete stop, using the stop deceleration SD. Bring the controller reference command to a complete stop, using the deceleration SD.

Table 3-22: **UM Values for Hard and Soft Stop**

Function 11: Abort motion

The behavior is similar to the Inhibit function with the exception that the "Abort" input release will not start the motor automatically. After the Abort is activated, MO=1 must be set either by communication or by the internal User Program.

The function activates the #@AUTO_ER routine, if it exists, in the user program.

Notes:

- Make sure that the drive you use has the actual digital input that is programmed. Failing to do so *will not* generate an error. Not all drives have the same digital input entries. Nevertheless no error indication would be given in case a "none existing" digital input is programmed. For example the Harmonica drive has 6 physical digital inputs. An attempt to set IL[10]=8 would not generate an error but the RLS function would be programmed to the drive.
- Use the Inhibit freewheel function with care. When the drive is shut, the motor applies no torque. Turning off a drive might leave the motor spinning until it stops by friction. In some situations this may be dangerous.
- When a switch is released, the attached function terminates. Functions 2, 3 and 4 (Full Stop, RLS and FLS) do not change the drive reference command. When the switch is released, the reference command (speed or position) is recovered. In order to ensure that reference recoveries do not generate discontinuities, the SD, VL[2] and VH[2] limits are used.

- IP and IB[N] can be used to detect a logically active switch of all defined functions, excluding function 2 ("No function is attached").

Attributes:	**Type**:	Parameter, Bit-field
	Assignment:	Yes
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default value:	IL[1]=0
		IL[2…10]=7 (RS), Non-volatile
	Range:	According to previous description
	Index range:	[1…10]
	Unit modes:	All
	Activation:	Immediate

Reference chapter in the *SimplIQ for Steppers Application Note:*
Chapter 12, "The Position Reference Generator"

See also:
UM, RM, JV, PX, BG, IP, IB[N], HM[N], HY[N]

IP - Input Port

Purpose:
Reports the state of digital input functions and the input pins.


Please note the difference between a pin state and input functions: Input pins are physical hardware. Input functions are an association, set by the IL[N], of hardware events to software events.

The low 16 bits of the IP report are the state of the digital input functions.
The high 16 bits report input pin states.

IP logic is always positive. When the digital input is *active*, the relevant IP bit is set.

The report is a bit-field, defined in the following table:

Bit	Description	Associated Function in IL[N] Command
0	General purpose input 1 is active	3
1	General purpose input 2 is active	3
2	General purpose input 3 is active	3
3	General purpose input 4 is active	3
4	General purpose input 5 is active	3
5	General purpose input 6 is active	3
6	Main home switch	8
7	Auxiliary home switch	9
8	Soft stop	7, 10
9	Hard stop	1, 10
10	Forward Limit (FLS)	5
11	Reverse Limit (RLS)	4
12	INH (Enable) switch	0
13	Hardware BG	6
14	Abort function	11
15	Not used; always 0	
16	Digital input 1 logical pin state	
17	Digital input 2 logical pin state	
18	Digital input 3 logical pin state	
19	Digital input 4 logical pin state	
20	Digital input 5 logical pin state	
21	Digital input 6 logical pin state	
22	Digital input 7 logical pin state	
23	Digital input 8 logical pin state	

Bit	Description	Associated Function in IL[N] Command
24	Digital input 9 logical pin state	
25	Digital input 10 logical pin state	
26 - 31	Reserved; always 0	

Table 3-23: **IP - Input Port**

Notes:

- Each type of SimplIQ for Steppers drive supports a different number of digital inputs. Please consult the drive's *Installation Guide* for more information about its inputs.
- For compatibility reasons inputs 7-10 do not have an indication for the "General Purpose" function and cannot be used for a user program AUTO routine as well. Bits 22-25 will still be set regardless of the above.
- IB[N] may be more convenient than IP for user program decisions and branching. However, it is not recommended for the synchronized reading of several input bits. If you need to read several bits of IP synchronously, use the IP command.

Attributes:

Type:	Status report, Integer
Source:	Program, RS-232, CANopen
Restrictions:	None
Unit modes:	All
SimplIQ:	Similar

See also:
IB[N], IL[N]

JV- Jogging Velocity

Purpose:
Sets the motor target speed for speed and position control (UM=2,3,4,5). After the next BG, the speed command is gradually changed to JV, according to the AC, DC and SF parameters.

In position-jogging mode (JV), and if the position feedback sensor is set to modulo counting (refer to XM[N] and YM[N]), a position-controlled motor can rotate forever. The position reading will jump each modulo count crossing, but the speed will remain steady.

Notes:

- Jog mode is recommended for homing procedures, because it does not require information about the starting position or destination.
- In position mode (UM=4, 5), a jogging command is under position control. The JV parameter determines the rate at which the position-command changes.
- In stepper mode (UM=3), JV determines the rate at which the electric angle command changes.
- In position control mode (UM=3, 4, 5), JV not only sets the speed of motion but it also states that the next motion will be a constant speed jog.
- For all relevant modes (UM=2, 4, 5), the value of JV must be set between VL[2] and VH[2]. Setting JV out of this range will invoke an "Out of limit" error code.

Attributes:		
	Type:	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	UM=2: None
		UM=3, 4, 5: MO=1
	Default values:	0 (RS), Non-volatile
	Range:	Velocity range
	Unit modes:	UM=2, 3, 4, 5
	Activation:	BG

See also:
AC, BG, DC, PX, XM[N], YM[N], SF, SP, VH[N], VL[N], MS

Application notes:
The Position Reference Generator

The Speed Reference Generator

KG[N] - Gain Scheduled Controller Parameters

Purpose:
Specifies the parameters of the gain scheduled speed or position controller. The KG[N] parameters apply only if the controller gains are scheduled, either manually (GS[2]=1...16) or automatically (GS[2]=64).

The following table details the use of the KG[N] parameters array:

Index	KG[N] Value	Length
[1…16]	KI for inner (speed) loop	16
[17…32]	KP for inner (speed) loop	16
[33…48]	KP for outer (position) loop	16

Table 3-24: **KG[N] Gain Scheduled Controller Parameters**

Attributes:	**Type:**	Parameter, [1…48]: Real;
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	RS=0
	Range:	[0 … 10^8]
	Index range:	[1…48]
	Activation:	Immediate
	SimplIQ:	Modified


Under most circumstances, this command is used only by the tuning environment.


Notes:

- The GS[N] array for SimplIQ for Steppers includes 16 gain scheduling controllers. In SimplIQ, it included 64 controllers.
- SimplIQ for Steppers cannot schedule the control filter, SimplIQ could schedule one link of the control filter.
- The units for the inner (speed) loop gains are:
  KP: mAmp/(count/sec)
  KI: mAmp/count
- The units for the position gain are 1/sec.

Application notes:
Gain scheduling

KI[N], KP[N] - PI Parameters

Purpose:

- KI[1], KP[1] defines the PI current control filter.
- KI[2], KP[2] defines the PI velocity control filter.
- KP[3] defines the gain of the position controller.

The parameters KP[2], KI[2] and KP[3] apply only if the controller gains are fixed (gain scheduling is not used: GS[2]=0).

 The units for the speed loop gains are:

KP: mAmp/(count/sec)

KI: mAmp/count

The units for position gain are 1/sec.

Attributes:	**Type:**	Parameter, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	RS=0
	Range:	KI[N]>0 KP[N]>0
	Index range:	[1…3]

See also:
KV[N], GS[N]

Reference chapters in the *SimpIIQ for Steppers* *Application Note:*
Chapter 10, "The Current Controller;" Chapter 15, "The Controller"

KL - Kill Motion and Program

Purpose:
Halts program execution and stops the motor. The KL command stops the execution of the user program and its automatic routines. It also issues the MO=0 motor disable command. KL freezes the status of the program and does not reset it. A later XC command will resume the program from the instruction at which the program was halted. Pending interrupts will remain pending.

A KL command issued when no program is running does nothing, and sets no error code.

Attributes:	**Type:**	Command, No value
	Source:	RS-232, CANopen
	Restrictions:	None
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

If the motor is on when KL is used, using XC to continue the program may fail, because the interrupted program expects the motor to be on, and may use commands restricted to the MO=1 state.

See also:
HP, XQ, XC

SimplIQ Programming and Language Guide

KV[N] - High-order Controller Filter Parameters

Purpose:
Specifies the parameters of the following filters:

Filter	Parameters	Maximum Order
Speed controller high-order filter	KV[0]…KV[21]	
Position controller high-order filter (RFFI)	KV[22]…KV[43]	
Analog position sensor filter	KV[44]…KV[55]	Order 4 (2 blocks)
Analog reference to speed controller	KV[56]…KV[67]	Order 4 (2 blocks)

Table 3-25: **KV[N] Filter Partitions**

The KV[0] parameter defines whether or not the speed controller high-order filter is used:

- If KV[0]=0, the high-order filter is not used.
- If KV[0]=100, the high-order filter is defined by the rest of the KV[N] parameters.

Similarly, KV[22]=0 deactivates the position controller high-order filter, KV[44]=0 deactivates the analog position sensor filter, and KV[56]=0 deactivates the filtering of the analog reference to the speed controller.

The KV[N] array specifies almost arbitrary linear filters. To learn how the KV[N] parameters specify a filter and to see a programming example, refer to Filters.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0
	Default values:	RS=0
	Range:	See the application note below
	Index range:	[0…67]
	Activation:	See note below
	SimplIQ:	Modified (see note)


Under most circumstances, this command is used only by the tuning environment.


Notes:

- The controller filters (speed and position) are activated by MO=1.

- The sensor and the analog speed reference filters are activated by writing KV[44] and KV[56] respectively.
- There is no restriction for the values written into the KV[N] array. The parameters are checked for consistency, however, upon activation. An inconsistent KV[N] set will prevent motor on by a BAD DATABASE exception.
- KV[N] are IQ25() representations of floating point numbers. For example, if we need to put the coefficient 0.75 into KV[2], we set KV[2]=round(2^{25}x0.75)= 25165824
- In SimplIQ the numbers of KV[N] had a very complex representation. Here they are just IQ25() numbers.

See also:
KI[N], KP[N]

Application notes:
Filters

The speed controller

LC - Current Limit Flag

Purpose:
Reports the status of the current limiting process.

After an excessive demand for large currents, the drive automatically limits itself to the continuous limit (CL[1]=1).

The LC flag returns the state of this automatic limiter.

Value	Description
0	The motor current is limited to the limit PL[1], or the motor is off.
1	The motor current is limited to the continuous limit CL[1].

Table 3-26: **LC Return Values**

Attributes:	**Type:**	Status report, Integer
	Scope:	Program, RS-232, CANopen
	Restrictions:	None
	Unit modes:	All
	SimplIQ:	Similar

See also:
MC, PL[N], CL[N]

Application note:
Current limiting

LD - Load Parameters from Flash

Purpose:
Loads all non-volatile variables from the flash memory to the RAM and resets all volatile variables to their default values.

Before accepting the loaded parameters, LD tests them as follows:

- The variables written in the flash memory can be read. The variables *cannot* be read if the drive is brand new and no parameters have ever been saved in it, or after a major firmware update.
- The variables in the flash memory are all in their permitted range. This test should not fail, as the legality of all variables is tested prior to saving.

If any of these tests fail, the contents of the flash are ignored and all non-volatile variables are set to their factory default (RS) states.

Certain exceptional variables are not reset by the LD command. The communication (PP[N]) parameters are retrieved from the flash memory, but are not set into action. The parameter PP[1] retains its old values in order to ensure communication continuity. The newly-loaded RS-232 communication parameters may be activated by PP[1]=1, and the new CAN parameters are activated by a "Communication Reset" NMT message.

The LD command may take a few milliseconds to perform, because it completely recalculates the drive database. At this time, the communication routines are disabled. If an LD command is executed by an RS-232 command, a CAN message may be lost in the meanwhile, and vice versa.

Attributes:		
	Type:	Command, No value
	Source:	RS-232, CANopen
	Restrictions:	MO=0, Program not running
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Modified – see note below

Notes:

- If an LD command fails, CD will report the reason for the failure by adding the string "Couldn't load from serial flash" followed by the reason for the failure.
- The Save() function used within a User Program does not check the integrity of the data before saving it to the flash memory.
- SimplIQ for Steppers difference: The LD command has no effect on the commutation and cogging correction tables stored in the motion processor. Refer to the SI command.

See also:
SV, SI,RS, CD

Application note:

Storing non volatile data

LL[N] - Low Feedback Limit

Purpose:

- *Speed limits*
  The LL[2] and HL[2] parameters define the limits of the allowed motor speed. If the motor speed exceeds HL[2] or is lower than LL[2], the drive is automatically disabled and the "Speed High Limit" fault (MF=0x20,000) is activated.
- *Position limits*
  LL[3] and HL[3] define the allowed motor position range for UM=3, 4 and 5. If the motor position is smaller than LL[3] or larger than HL[3], the motor is automatically disabled and the "Position High Limit" fault (MF=0x400000) is activated. In order to re-enable the motor, modify the current position (PX) within the ranges of HL[3] and LL[3].

Attributes:		
	Type:	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0 and For LL[2] and HL[2]: HL[2]>LL[2] For LL[3] and HL[3]: HL[3] >LL[3]
	Default values:	LL[2]=-1,000,000 HL[2]=1,000,000 LL[3]=-15,000,000,000 HL[3]=15,000,000,000 (RS), Non-volatile
	Range:	LL[2]: Velocity range HL[2]: Velocity range $-2^{31} \leq LL[3] \leq 2^{31} - 1$ $-2^{31} \leq HL[3] \leq 2^{31} - 1$
	Index range:	[2, 3]
	Unit modes:	HL[2], LL[2]: UM=2, 3, 4, 5 HL[3], LL[3]: UM=3, 4, 5
	Activation:	Immediate
	SimplIQ:	Similar

Note: The position counter counts modulo; refer to position counter range.

See also:
VL[N], VH[N], MF, SR, MO, XM[N]

LP[N] - List Properties

Purpose:
Sets the properties of the non-volatile data upload and download by the next LS and DL commands.

- LP[1] sets the start byte address of the next LS transmission, or the byte address for starting the storage of the next DL transmission.
- LP[2] sets the size — in bytes — to be transmitted at the next LS.
- LP[3] returns the start byte address of the user program (read only).
- LP[4] returns the size of the user program storage (read only).

Attributes:	**Type:**	Parameter, Integer
	Scope:	RS-232, CANopen
	Restrictions:	None
	Default values:	LP[1] and LP[2] set to 0 on power on
	Range:	LP[1]: [0…262,144] LP[2]: [0…247]
	Index range:	[1…4]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar


Under most circumstances, this command is used only by the Elmo Studio.

See also:
CC, DL, LS

SimplIQ Programming and Language Guide

LS - List User Program

Purpose:
Uploads non-volatile data from the drive to the host, according to the parameters of LP[N]. The most common use of LS is to retrieve the user program and to retrieve the personality data of the drive.

LS begins to send data from the byte address of LP[1] in the flash memory. The length of the transmitted data is LP[2] bytes.

The format of the LS message is:
[data payload][esc][checksum]
where:

- The data payload is in the hex-binary format.
- esc is the 0x1b (27 decimal) character.
- The checksum is calculated for 16 bits in 2's complements.


Use the LS command with care because it is not entirely "safe": while the program listing uploads to the communication lines, no program instructions are executed, and no communicated commands are interpreted.

Attributes:	**Type:**	Command, No value
	Source:	RS-232
	Restrictions:	None
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar


Under most circumstances, this command is used only by the Elmo Studio.

See also:
CC, DL, LP[N], LS

SimplIQ Programming and Language Guide

MC - Maximum Peak Driver Current

Purpose:
Reports the maximum phase current allowed for the drive, in amperes. This command informs the software about the type of servo drive used with the controller.

Attributes:	**Type:**	Report, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	Read only
	Unit modes:	All
	SimplIQ:	Similar

You may limit the current for a specific application using the PL[1] and CL[1] commands.

See also:
IQ, ID, CL[N], PL[N]

MF - Motor Failure

Purpose:
Reports the reason why the motor has been automatically shut down (set to MO=0). MF normally reports zero (as default). The fact that the motor has been automatically shut down is reflected as a bit in the status register (SR) report, and MF provides the detailed information.

After a fault, the MF value remains fixed, even if the reason for the fault no longer exists. MF is automatically set to zero on the next motor enable MO=1.

Fatal faults (CPU exceptions) are permanent and can be resent only by power reset.

The following table lists the type of faults reported by the MF value.

Reported Fault	Value	Bit
The motor is on, or the last motor shutdown was the normal result of a software command.	0	
Sensor problem: The amplitude of the analog sensor is out of range (see PF[18],PF[19]).	0x1	0
Encoder #2 error	0x2	1
Feedback loss: no match between encoder and Hall location. Available in encoder + Hall feedback systems.	0x4	2
The peak current has been exceeded. Possible reasons are drive malfunction or bad tuning of the current controller.	0x8	3
Inhibit.	0x10	4
Motion processor could not detect SimplIQ for Steppers activity	0x20	5
Two digital Hall sensors were changed at the same time. Error occurs because digital Hall sensors must be changed one at a time.	0x40	6
Speed tracking error DV[2] - VX (for UM=2 or UM=4, 5) exceeded speed error limit ER[2]. This may occur due to: • Bad tuning of the speed controller. • Too tight a speed error tolerance. • Inability of motor to accelerate to the required speed due to too low a line voltage or a motor that is not powerful enough.	0x80	7

Reported Fault	Value	Bit
Position tracking error DV[3] - PX (UM=5) or DV[3] - PY (UM=4) exceeded position error limit ER[3]. This may occur due to: • Bad tuning of the position or speed controller. • Too tight a position error tolerance. • Abnormal motor load, or reaching a mechanical limit.	0x100	8
Cannot start because of inconsistent database. The type of database inconsistency is reflected in the SR status report, and in the CD CPU dump report.	0x200	9
Too large a difference in ECAM table.	0x400	10
Heartbeat failure. Error occurs only if drive is set to abort under heartbeat failure in a CANopen network (object 0x6007 in CAN object dictionary is set to 2).	0x800	11
Servo drive fault. Error described according to the servo drive fault detail bits 13 - 15 in the MF report. Refer to following table.	0x1000	12
Servo drive fault detail bit 1. Refer to following table.	0x2000	13
Servo drive fault detail bit 2. Refer to following table.	0x4000	14
Servo drive fault detail bit 3. Refer to following table.	0x8000	15
Failed to find the electrical zero of the motor in an attempt to start it with an incremental encoder and no digital Hall sensors. The reason may be that the applied motor current did not suffice for moving the motor from its position.	0x10,000	16
Speed limit exceeded: VX<LL[2] or VX>HL[2].	0x20,000	17
Stack overflow - fatal exception. This may occur if the CPU was subject to a load that it could not handle. Such a situation can arise only due to a software bug in the drive. Use the CD command to get the CPU dump and report to your service center.	0x40,000	18
CPU exception - fatal exception. Something such as an attempt to divide in zero or another fatal firmware error has occurred. Use the CD command to get the CPU dump and report to your service center.	0x80,000	19
Timing error (internal software problem)	0x100,000	20
Motor stuck - the motor is powered but is not moving according to the definition of CL[2] and CL[3].	0x200,000	21
Position limit exceeded: PX<LL[3] or PX>HL[3] (UM=5), or PY<LL[3] or PY>HL[3] (UM=4).	0x400,000	22

Reported Fault	Value	Bit
Motion processor inhibited (internal software problem)	0x800,000	23
Cannot auto-calibrate current offsets.	0x10,000,000	28
Cannot start motor.	0x20,000,000	29
Modulo limit exceeded (internal software problem)	0x80,000,000	31

Table 3-27: **Reasons for Automatic Motor Shutdown**

0x8000	0x4000	0x2000	Meaning
0	0	0	OK.
0	0	1	Under-voltage. The power supply is shut down or it has too high an output impedance.
0	1	0	Over-voltage. The voltage of the power supply is too high, or the servo drive did not succeed in absorbing the kinetic energy while braking a load. A shunt resistor may be required.
0	1	1	Reserved.
1	0	0	Reserved.
1	0	1	Short circuit. The motor or its wiring may be defective, or the drive is faulty.
1	1	0	Temperature. Drive overheating. The environment is too hot, or lacks heat removal. There may be a large thermal resistance between the drive and its mounting.
1	1	1	Reserved.

Table 3-28: **Bits 13 - 15 of MF Report**

Attributes:

Type: Status report, Integer, Bit-field
Scope: Program, RS-232, CANopen
Restrictions: None
Unit modes: All
SimplIQ: Modified – see notes

Notes:

- In SimplIQ for Steppers the user can set amplitude tolerances for analog encoder.
- Some internal problems are reported in SimplIQ for Steppers that are absent from SimplIQ.

Example:
The MF report of 0x3000 indicates that the motor has been shut down due to under-voltage. The under-voltage condition did not necessarily exist at the time the MF was reported. It may possibly be the result of high power consumption and a high output impedance of the power supply. In such a case, when the motor was shut down the power consumption stopped and the power supply returned to its normal voltage.

See also:
SR, MO, TS, LL[N], HL[N], CD, PE, VE

MI - Mask Interrupt

Purpose:
Selects which interrupts (automatic routines) are active.

A user program may include a main code section and some automatic routines. When the program runs, the conditions for calling these routines are checked continuously. If the conditions for running an automatic routine[3] are met, it is called. At certain times, you may want to block some of the automatic routines. For example:

- An #@AUTO_RLS automatic routine may be deactivated in a homing process.
- You may want a certain code sequence to be non-interruptible.
- Certain auto-routines may be needed only when starting the program from certain labels.

MI is a bit field. Each bit in MI masks its corresponding automatic routine, preventing its execution. The MI bits are detailed in the following table:

MI Value	Masked Interrupt	Relevant Routine
1 (0x1)	Not used	0
2 (0x2)	Abort	AUTO_ER
4 (0x4)	Soft stop	AUTO_STOP
8 (0x8)	Soft begin	AUTO_BG
16 (0x10)	RLS	AUTO_RLS
32 (0x20)	FLS	AUTO_FLS
64 (0x40)	Switch enable	AUTO_ENA
128 (0x80)	Digital input 1	AUTO_I1
256 (0x100)	Digital input 2	AUTO_I2
512 (0x200)	Digital input 3	AUTO_I3
1024 (0x400)	Digital input 4	AUTO_I4
2048 (0x800)	Digital input 5	AUTO_I5
4096 (0x1000)	Digital input 6	AUTO_I6
8192 (0x2000)	Main home event	AUTO_HM
16,384 (0x4000)	Auxiliary home event	AUTO_HY
32,768 (0x8000)	User program error	AUTO_PERR

Table 3-29: **MI Bits**

[3] These conditions include the requirements that no interrupt request of high priority is pending (in which case the corresponding automatic routine enters the pending list) and that the interrupt is not masked (in which case the interrupt is ignored).

Notes:

- MI is not affected by the XQ command. You should set MI to the desired value in the first lines of your user program in order to ensure that the correct automatic routines can run.
- Several interrupts may be blocked by the MI command. MI=65,535 or MI=0xffff will block all interrupts of the table. MI=48 will block only the AUTO_FLS and the AUTO_RLS interrupts. MI=MI|2 will block AUTO_ER and leave all other interrupt mask bits as is.
- If AUTO_PERR is activated, all other interrupts will be masked (MI=0x7fff).

Attributes:		
	Type:	Parameter, Integer, Bit-field
	Scope:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	0, Volatile
	Range:	[0…65,535]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

See also:
XQ, XC

SimplIQ Programming and Language Guide

MO - Motor Enable / Disable

Purpose:
Enables and disables (freewheels) the motor power.

- *Disabling the motor*
  MO=0 disables the motor. This is the idle state of the drive. The power stage is disabled and no current flows in the motor. In this mode the servo drive can perform various tasks that are impossible when the motor is on:
  - Boot on power up.
  - Calculate and check the integrity of the drive database.
  - Download new firmware and user programs.
  - Save parameters in the flash memory.
  - Modify setup data that cannot be modified on-the-fly, such as commutation parameters (CA[N]) and unit mode (UM).

  The servo driver is automatically disabled when a motor fault is captured (MF). An attempt to enable the motor may fail if the conditions of the fault still exist.

  If the brake function is activated, the MO=0 command duration is extended according to the BP[N] command specification.

- *Enabling the motor*
  MO=1 is the operative state of the servo drive, driving the motor and activating and executing the programmed motion. The software runs a set of tests to ensure that all conditions for running the motor are met.

  If MO is set to 1 and the motor is already on, nothing happens.

  When the motor is enabled, the drive reinitializes the internal parameters and motion drivers. The drive may fail to start if the setup data is found to be inconsistent (for example, CA[4]=CA[5]). In this case, CD commands indicate the reason for the failure.

  The last captured motor fault (MF) is reset to zero.

  In the position control modes (UM=4, 5), the motor is always started so that it does not jump. The total position control command — which consists of the internal position command and the external position command — is set to the actual present position of the motor in order to prevent the motor from jumping.

Notes:

- When RM=1 the servo drive will attempt to start (set MO=1) automatically after power on. A servo drive that has been shut down for any reason will attempt an automatic restart every few milliseconds. Automatic restart will occur only if the Enable switch (INH) is active (refer to the IL[N] command). In all other modes, the MO=0 state is maintained until MO is explicitly changed by a software command.

- In encoder-only systems (in which no digital Hall sensors are present), the commutation is calculated only once after power up upon the first MO=1 command. The motor moves a few encoder counts during the automatic commutation search.
- When the brake function is enabled, the dedicated output is released after the duration defined in BP[N]. During this time, all motion reference commands are ignored.
- After a motor fault (MF>0), the motor can be re-enabled only when the cause of the fault is no longer present.
- If MO=1 fails with the "Amplifier not ready" error code (EC=66), the exact reason for the fault can be found by querying the MF command.
- MO=0 disables the motor for 200 sampling times (about 10 milliseconds) until a new MO=1 can restart the motor.
- In the "Ready to switch on", "Switch on disabled" and "Fault" states (refer to the Elmo *CANopen Implementation Manual*), MO=1 is blocked and returns an error. These states can be command controlled only by a CAN master using the DS402 standard control word (object 0x6040).
- Setting MO=1 in the "Switched on" CAN state transfers the state of the state machine to "Operation enabled." Setting MO=0 in "Operation Enabled" or "Quick stop active" state will transfer the state machine to "Switched on." (Refer to CAN object 0x6040 in the Elmo *CANopen Implementation Manual*).

Attributes:		
	Type:	Command/Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	0, Volatile
	Range:	[0…1]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

See also:
MF, SR, CD, BP[N]

MP[N] - Motion (PT/PVT) Parameters

Purpose:
Programs the parameters of PVT or PT motions. PT or PVT motion is programmed as a sequence of points that are visited at programmed times.

The QP[N] array stores the position reference points used for PT or PVT motion. The QV[N] and QT[N] arrays store the speed and timing data that is additionally required for PVT motions.

The entries of the QP[N], QV[N] and QT[N] arrays that are not used for an executing motion may be programmed, in order to enable part of an array to be used for running the motion while another part of the array is programmed for subsequent motions.

The QP[N], QV[N] and QT[N] arrays may be referred to as cyclical buffers. In cyclical mode, periodic motions can be set to run forever. A host may program the table on-the-fly, generating an infinite, online updated motion.

The MP[N] array defines how the QP[N], QV[N] and QT[N] tables are used for PVT and PT motions, as detailed in the following table.

Parameter	Description
MP[1]	First index in the QP[N], QV[N] and QT[N] arrays to be used for the motion.
MP[2]	Last index in the QP[N], QV[N] and QT[N] arrays to be used for the motion.
MP[3]	0: The motion is to terminate after the last (MP[2]) element of the QP[N], so QV[N] and QT[N] arrays are used. A PT or a PVT motion terminates by decelerating the motor to a complete stop, using the SD deceleration. When using CAN, an emergency object is transmitted. 1: The motion is to be cyclical, so after using the last (MP[2]) element of the QP[N], QV[N] and QT[N] arrays, the first (MP[1]) element is used again. 2: Reserved 3: CAN is being used and the motion is to be terminated without an emergency object.
MP[4]	Used for PT motions only. The number of drive sampling times between consecutive specifications of position reference points. Note that MP[4] counts sampling times for the position controller. This sampling time is given by WS[28].
MP[5]	If CANopen communication is used, allows the drive to send an emergency object to the host when only a predefined number of valid reference points has been left in the QP[N], QV[N] and QT[N] motion arrays. This way, the host is released from polling the status continuously for the motion queue. MP[5] programs the number of valid motion points remaining when an emergency object is sent to the host. If MP[5]=0, no emergency object is sent.

Parameter	Description
MP[6]	Reports the next entry index (write pointer) for the following point in the PVT/PT table. This report is required for running PVT and PT motions using CANopen and special high-speed motion referencing methods that are available only for CANopen.

Table 3-30: **MP[N] Parameters**

MP[1] and MP[2] cannot be changed during a PT or PVT motion.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	**PVT:** $1 \leq MP[1] < MP[2] \leq 64$ **PT:** $1 \leq MP[1] < MP[2] \leq 1024$ **General:** $MP[2]-MP[1] \geq 1$
	Default values:	MP[1]=1, MP[2]=64, MP[3]=1, MP[4]=4, MP[5]=50, MP[6]=1, Volatile
	Range:	**PVT:** MP[1]: [1…64] MP[2] : [1…64] MP[3] : [0…3] MP[4] : [1…256] MP[5] : [0…62] MP[6] : [1…64] **PT:** MP[1] : [1…1024] MP[2] : [1…1024] MP[3] : [0…3] MP[4] : [1…256] MP[5] : [0…62] MP[6] : [1…1024]
	Index range:	[1…6]
	Unit modes:	UM=3, 4, 5
	Activation:	Immediate
	SimplIQ:	Similar

See also:
PT, PV

Application note:
The Position Software Reference Generator

MS – Motion Status

Purpose:
Reports the status of the motion profiling process. MS can be used for detecting the end of motions: a PTP motion that has reached its target, or a completed PT or PVT motion.

- For **position control modes** (UM=3, 4, 5), MS reports as follows:

Value	Description
0	Motor position stabilized. The *feedback* position is steady within the ranges defined by TR. Note that value 0 is applicable only when there is a defined position target, as in PTP.
1	The *reference* for the position controller is stationary, or the motor is off (MO=0).
2	The *reference* to the position controller is dynamically controlled by one of the optional motion profilers: PTP, Jog, PT or PVT.
3	Reserved.

MS reflects only the software motions. It does not indicate anything about the behavior of the external reference command.

- For **speed control modes** (UM=2), MS reports as follows:

Value	Description
0	Not applicable for this mode.
1	▪ The *reference* to the speed controller equals the speed target. (This is always the case if no profiler mode is used: PM=0.) ▪ The motor is off (MO=0).
2	The software *reference* to the speed controller differs from the speed target. In software profiled reference mode (PM=1, RM=0), this is the case while accelerating, decelerating or smoothing to the JV target speed.
3	Reserved.

Attributes:

Type:	Status report, Integer
Scope:	Program, RS-232, CANopen
Restrictions:	None
Unit modes:	All
SimplIQ:	Similar

See also:
PM, RM, DV[N], AC, DC, TR

Application note:
Idle Mode and Motion Status

OB[N] - Output Bits Array

Purpose:
Sets and resets an output bit. The OB[N] command only sets a digital output that is defined by OL[N] as a general purpose output.

- OB[N] for N=1…6 returns the value of digital output (N), if digital output N is defined as general purpose output. Otherwise, it returns 0.
- OB[10…15] are reserved.
- If at least one of the digital outputs is mapped to the Amplifier OK function, OB[7] returns 1 if the drive is ready (MO=1, or MO=0 and no exception such as under-voltage prevents MO=1), or 0 if the drive is not ready. If none of the digital outputs is mapped to the Amplifier OK function, OB[7] returns 0.
- If at least one of the digital outputs is mapped to the Brake function, OB[8] returns 1 if the brake is engaged, or 0 if the brake is released. If none of the digital outputs is mapped to the Brake function, OB[8] returns 0.
- If at least one of the digital outputs is mapped to the Motor enable/disable function, OB[9] returns 1 if the MO=1, or 0 if the MO=0. If none of the digital outputs is mapped to the Motor enable/disable function, OB[9] returns 0.
- For unused bits (N=[10…15]), OB[N] returns 0.

Notes:

- Each of the SimplIQ drives supports a different number of digital outputs. The number of digital outputs is specified in the drive's *Installation Guide*. The value of OB[N] varies according to the logic state of the output even if the output does not exist.
- The OB[N] syntax may be more convenient than OP for setting individual outputs. However, it is not appropriate for the synchronized setting of several output bits.
- OB[N] will not affect digital outputs defined as AOK and/or Brake. If OB[N] is applied to an output that is defined for a specific function, no error indication will be sent, but the command will do nothing.
- OB[N] sets and reads the logical values of the outputs. The voltage values at the digital output connector pins are active high or active low with respect to OB[N], according to the OL[N] setting.
- At boot, all OB[N] values not defined for the Amplifier OK function are set so that the output opto-couplers do not conduct. This way, no transition will occur at the digital outputs when the drive boots. If another value is required immediately after boot, use an #AUTOEXEC routine.
- Since Output Compare is "hard wired" to digital output #1, any setting or reading of digital output #1 will be false when Output Compare is active (because the pins are shared and Output Compare signals take precedence).

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default value:	0 (RS), Volatile
	Range:	[0, 1]
	Index range:	[1, 6] (Set); [1…15] (Get)
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

See also:
OP, OL[N], BP[N]

OC[N] – Output Compare

Purpose:
Output a signal when the present position is compared to a user defined position.

The OC command generates a train of pulses according to the encoder position values.

The OC[1] command operates in two modes:

1. OC[1]=1: Absolute position mode. The first pulse is generated by the initialized absolute position defined by OC[2] (i.e. PX=OC[2] or PY=OC[2] depending on the OC[6] setting) and continues at position intervals specified by the OC[3] value. That is, PX or PY =OC[2]+k*OC[3], where k is the number of successful compared occurrences (k=1,2,...).
2. OC[1]=2: Immediate mode. The first pulse is generated immediately after the command is received by the drive (no initialized absolute position value is required) and continues at position intervals specified by OC[3] counts.

The number of occurrences in which the pulses are generated can be limited by OC[5]. The duration of a pulse can be set by OC[4].

Index	Description
1	0: Disable Output Compare 1: Accept last changes in OC[N] and enable Output Compare beginning at Absolute Position OC[2]. Note: to get proper execution of the command, the following condition should be met: OC[3]*(OC[2]-PX)>0 when the source is the main feedback, or OC[3]*(OC[2]-PY)>0 when the source is the auxiliary feedback. 2: Accept last changes in OC[N] and enable Output Compare immediately
2	The absolute position for first pulse (PX or PY value). Applicable only in absolute position mode. This value cannot exceed the modulo limit in the same direction of motion i.e. |OC[2]|-|PX| or |OC[2]|-|PY| must be positive.
3	The position interval between subsequent pulses (in encoder counts). The positive/negative value of OC[3] should be set according to the direction of the encoder motion. When the direction is positive (increasing PX value) OC[3] should be positive; otherwise it should be negative. Note that even for one output pulse OC[3] is important, because it indicates the motion direction in which the pulse will emit.
4	N: Pulse duration in 25*(N+1) [nSec].
5	N: Number of pulses to generate 0: Infinite Output Compare mode (train of pulses will end only with the OC[1]=0 command)

Index	Description
6	Output Compare Source Signal
	0: Output Compare on Main feedback
	1: Output Compare on Auxiliary feedback
	2: Output Compare on Position command

OC[1] returns the following values:

-3 Referenced-based comparison has been aborted because the motor has been shut down.

-2 No more pulses are being generated because the maximum rate has been exceeded.

-1: No more pulses are being generated because the number of pulses specified in OC[5] has been reached.

0: Output Compare function is disabled.

1: a. Output Compare at absolute position:

Output Compare function has started but absolute position has not yet been reached; therefore, the train of pulses has not begun.

b. Immediate Output Compare:

Output Compare function has started but first pulse has not yet been produced.

2: The train of pulses is being generated now.

Notes:

1. For output compare to work on an auxiliary sensor, it must be set up as a position sensor input.
2. The Output Compare pulses cannot exceed 10 kHz. Larger rates will cause OC to abort (OC[1] will return -2).
3. Note that long duration pulses may cause a number of pulses to connect and produce one overlapping pulse. The drive gives no warning on such occasions.
4. When going in the direction that is opposite to the specified direction with the same OC[3] value, the compare will not occur.
5. When Output Compare is active, Digital Output #1 is used to produce the train of pulses. The configuration of this output, as set by OL[1], OB[1] or OP, is ignored.
6. OC[2] = 2 (comparison on position command) is possible only if the motor is on and the unit mode is 3, 4 or 5. If OC[2] = 2, a motor shut down will also cause the pulse comparison to be aborted.
7. In order to prevent output of an additional pulse during the activation or deactivation of the feature, it is recommend to configure the high logic level for Output #1 (set OL[1]=1) before the OC[] activation.
8. If a setting of OC[1] fails, it still kills any ongoing output compare process.

9. Changing the position counter (by changing PX or PY, or by an active homing process) may lead the "Output Compare" function to generate unpredictable results. Note that the software does not check for this and it is the user's responsibility to avoid setting the position counters while "Output Compare" is active.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	OC[1]…OC[2], OC[5]…OC[6]=0 , OC[3]=100(RS), volatile OC[4]=200(RS), volatile
	Range:	OC[1]: [0…2] OC[2]: [-10^9 … 10^9] OC[3]: [-10^9 … 10^9] OC[4]: [1…32,767] OC[5]: [0…10^9] OC[6]: [0..2]
	Index range:	[1…6]
	Unit modes:	All
	Activation:	After OC[1]=1 or OC[1]=2
	SimplIQ:	Modified

Notes:

- In SimplIQ for Steppers output compare performs similarly on all the sensor types. There is no difference between incremental encoders and other sensors.
- **SimplIQ differences:**
  - The frequency limit is 10 kHz. If the frequency restriction is violated, the amplifier might not crash, but pulse comparison may be aborted.
  - No stray pulses appear at back rotation.
  - Homing can run normally while output compare is on.
  - Compare distance (OC[3]) is not limited to ±32767
  - The compare distance (OC[3]) may be ±1.
  - The number of issued pulses is not limited to 65535.

See also: YA

OL[N] - Output Logic

Purpose:
Defines the logic level and function behavior of the digital outputs. The drive has several non-committed digital outputs. Each of these outputs can be programmed to a specific function and logic level.

Each OL[N] entry is dedicated to a certain output: OL[1] for the DOUT1 connector pin and OL[2] for the DOUT2 connector pin, … OL[N] for the N connector pin

The following table summarizes the functions that can be attached to the digital output pin.

OL[N] Bits	Meaning	Values
0	Logic levels	0: Low level active.
		1: High level active.
1 - 4		0: General purpose. Serves as a simple digital output function, subject to the OB[N] or OP setting.
		1: AOK, drive ready for use.
		2: Brake. Responds according to definition in BP[N].
		3: Motor enable/disable indication.
		4: Reserved.
5 - 15	Reserved	

Table 3-31: **OL[N] Functions**

Terms:

- Logic level low:
  When the function is active, the opto-coupler is conducting.
- Logic level high:
  When the function is active, the opto-coupler is open circuit.
- AOK:
  Indicates that the drive is ready. The AOK signals that the physical conditions needed to run the motor are available. If AOK is flagged, the motor DC voltage is within range, the drive temperature is good, and no over-current or short-circuit has been captured during the previous 10 milliseconds. When programming this option, OB[N] and OP have no influence on the relevant output.
- Brake:
  The output is mapped to brake functionality as defined in BP[N]. When programming this option, OB[N] and OP have no influence on the relevant output. The brake function works only in positive logic.
- Motor Enable/disable

  The output is mapped to this function as defined in the MO command. When programming this option, OB[N] and OP have no influence on the relevant output.

The possible values of OL[N] are outlined in the following table.

Command Value	Active Level	When Active . . .
OL[N]=0	Low	Output serves as general purpose.
OL[N]=1	High	Output serves as general purpose.
OL[N]=2	Low	AOK: drive ready for use.
OL[N]=3	High	AOK: drive ready for use.
OL[N]=4	Low	Brake feature is active.
OL[N]=5	High	Reserved.
OL[N]=6	Low	Motor enable/disable indication.
OL[N]=7	High	Motor enable/disable indication.

Table 3-32: **Possible Values for OL[N]**

Notes:

- The number of outputs in each type of *SimpIIQ for Steppers* drive differs. Nevertheless there will be no error indication when setting an output that does not physically exist on the drive. For example the Harmonica drive will receive the command OL[6]=3 with no error. As output #6 does not exist in the Harmonica no AOK indication can be retrieved from the drive.
- Since the Output Compare output is "hard wired" to physical digital output #1, any configurations of this output (OL[1]) are ignored when the feature is active.

Attributes:

Type:	Parameter, Integer, Bit-field
Process:	Yes
Assignment:	Yes
Scope:	Program, RS-232, CANopen
Restrictions:	None
Default values:	0, Non-volatile
Range:	According to basic logic table
Index range:	[1, 6]
Unit modes:	All
Activation:	Immediate
SimpIIQ:	Similar

See also:
OB[N], OP, BP[N], MO, OC[N]

OP - Output Port

Purpose:
Sets values for all uncommitted digital outputs, defined as *general purpose* by the OL[N] command. OP does *not* affect the digital output pins that are otherwise defined. The bits of OP[N] can be individually accessed by OB[N]. For more information, refer to the OB[N] and OL[N] commands.

When OP is queried, the status of the "Amplifier OK" , "Brake" and "Motor enable/disable" functions are reflected in bits 7, 8 and 9 respectively.

Notes:

- The number of outputs in each type of *SimpIIQ for Steppers* drive differs. Accordingly, the outcome of this command may differ between drives. Please consult the drive's *Installation Guide* for more details about the number of its digital outputs..
- When any of the uncommitted digital outputs are defined as *general purpose*, the physical state of the output depends on the previous OP command setting and its logic level defined by the OL command.
- The OB[N] syntax may be more convenient than OP for setting individual outputs. However, it is not appropriate for the synchronized setting of several output bits. If a synchronized setting of several digital outputs is desired, use the OP command.
- Since Output Compare is "hard-wired" to physical digital output #1, any setting or reading of this output will be false when the feature is active.

Attributes:	**Type:**	Parameter - Status, Integer, Bit-field
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	0, Volatile
	Range:	[0…63] for write For querying, as described above
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

See also:
OB[N], OL[N], BP[N], OC[N]

PA - Absolute Position

Purpose:
Specifies that the next software position command will be a PTP (point-to-point) and defines the target position for the next PTP motion. The position reference to the drive is composed of an "internal" software command and an external command, calculated from the analog inputs and the auxiliary feedback input.

In PTP motion, the drive calculates the software position reference so that a desired target position will be reached as fast as possible, subject to the speed (SP) and the acceleration limits (AC, DC, SD) (with possible smoothing: SF). A PTP command can be given at any time, at any speed, regardless of the present executing motion. The drive calculates the acceleration, speed and minimum time path to the target position, starting from the present position and speed.

A PTP motion, like any other software motion, is initiated by a BG command. For example, a PA=1000 command specifies that the next BG command will activate a PTP motion, overriding any previous motion specification, and that the target position will be 1000. The new PA value causes PR to become 0. In order to make sequential PTP movements of the same size, use the PR command. Subsequent PR settings may cause the PA value to be changed.

Notes:

- The PA value must be within the modulo range: (XM[1]…XM[2] - 1) for UM=3, 5 or (YM[1]…YM[2] - 1) for UM4, and restricted by the VL[3] and VH[3] settings.
- Modulo calculation is always active; therefore, PTP motions are planned using the shortest way. For example, if XM[1]=-500, XM[2]=500, UM=5, the present position command is 490 and the next PA is set to -490, the PTP motion will be planned in the positive direction, and the resulting motion length will be 20 counts.
- A PTP motion will continue until completion even if the position counter value has been reset on-the-fly (refer to the HM[N] / HY[N][4] commands). However, the position target does not change so that in the case of PA=n, the BG sequence may result in a different motion length than initially programmed.

Attributes:		
	Type:	Command/Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=1
	Range:	Position counter range
	Unit modes:	UM=3, 4, 5
	Activation:	BG
	SimplIQ:	Similar

See also:
PR, BG, MO, XM[N], YM[N], MS, VH[N], VL[N], SP, AC, DC, SD, SF, HL[N], LL[N]

[4] In dual loop mode (UM=4), position feedback is PY. It can be manipulated on-the-fly by HY.

Application note:
The Position Reference Generator

PE - Position Error

Purpose:
Returns the present position tracking error, in counts.

- In *main* feedback position mode (UM=5), PE reads:
  PE = DV[3] – PX.
  PE is read modulo-XM[N], taken the shortest way.
  For example, if XM[1]=-500, XM[2]=500, DV[3]=400 and PX=-400, PE will read 200.
- In *auxiliary* feedback position mode (UM=4), PE reads:
  PE = DV[3] – PY.
  PE is read modulo-YM[N], taken the shortest way.
  For example, if YM[1]=-500, YM[2]=500, DV[3]=400 and PY=-400, PE will read 200.

If the absolute value of PE exceeds ER[3], motion is aborted and the motion fault code MF=256 (0x100) is set. If MO=0, or if the position controller is not used (UM=1, 2 or 3), PE returns 0.

Attributes:		
	Type:	Status report, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Unit modes:	4, 5
	SimplIQ:	Similar

See also:
XM[N], YM[N], ER[N], MF, UM

Application note:

The position controller

PK - Peek Memory

Purpose:
Returns the DSP memory dump for an address range.

Syntax:
PK=N
where N is a 32-bit number whose least significant 24 bits contain the starting DSP memory address and the most significant eight bits contain the data length. The data length is limited to 128 words, due to the limitation of the output buffer. If the data length is 0, 128 or greater, the PK command returns 128 words by default. When the data length equals 2, the *SimplIQ* drive regards the data as a long variable and copies it to the output buffer in the critical section, in order to ensure its integrity.


Under most circumstances, this command is used only by the Elmo Studio.

Example:
PK=0x7f000200 returns the contents of the DSP memory starting at address 0x200 and continuing through word 0x280.

Attributes:	**Type:**	Status report, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Range:	[0…0x80FFFFFF]
	Unit modes:	All
	SimplIQ:	Similar

See also:
CD

SimplIQ Programming and Language Guide

PF[N] – Floating point parameters

Purpose:

This parameter vector is used for device data, which is normally entered by the tuning environment. The user is encouraged to use the tuning environment instead of entering these parameters manually.

Parameter	Description	Units [range]	Default
PF[1]	Motor induction. Used for correcting the motor voltage at high speeds.	mH. [0.05 … 500]	1.0
PF[2]	Motor torque constant, Ke. Used for correcting the motor voltage at high speeds.	Volt/(Rad/Sec), [0 ...50]	0.1
PF[3]	Current loop delay, used for cogging compensation	Sec, [0 ... 2e-3]	1e-4
PF[4]	Maximum electrical frequency for applying full cogging correction. Beyond this frequency, cogging correction decreases.	Hz, [80 ... 5000]	200
PF[5]	Maximum electrical frequency for applying cogging correction at all. Beyond this frequency, cogging correction is not made. For proper operation, keep PF[5]>PF[4].	Hz, [80 ... 5000]	450
PF[6]	Reserved		40
PF[7]	Ratio between current command step size and resulting current overshoot. Serves in overshoot limiting of the current loop.	Pure number, [0…1]	0.20
PF[8]	Ratio between current command slope in (Unit/Ts) and resulting current overshoot. Serves in overshoot limiting of the current loop.	Pure number [0…3]	0.15

Parameter	Description	Units [range]	Default
PF[9]	High frequency filter corner for the current controller additional filter.	Hz, [100…10000] PF[9]<100 means "don't use filter"	0
PF[10]	High frequency filter corner for the analog input #1 software filter.	Hz, [100 …10000] PF[10]<100 means "don't use filter"	0
PF[11]	High frequency filter corner for the analog input #2 software filter.	Hz, [100 …10000] PF[11]<100 means "don't use filter"	0
PF[12]	DC bus voltage filter cutoff frequency	Hz, [100 …10000] PF[12]<100 means "don't use filter"	500
PF[13]	Input filter for current loop, cutoff frequency	Hz, [100 …10000] PF[13]<100 means "don't use filter"	2000
PF[14]	Maximal step for current command	Ampere, [0..MC]	MC
PF[15]	Over-voltage – voltage > PF[15] will cause an over-voltage exception. The under-voltage exception is operated at voltages less than PF[15]/2	Volts, [8..BV]	BV
PF[16]	Moment of inertia of motor + load inertia reflected to motor	Kg M^2 , [1e-8…100]	1e-5
PF[17]	Damping of notch of position reference on open loop stepper drive	Pure number, [0.05 … 0.8]	0.1
PF[18]	Minimum threshold, in Volts, for analog encoder p-p signal	Volts, [0..1.5]	0
PF[19]	Maximum threshold, in Volts, for analog encoder p-p signal	Volts, [0..1.5] 0: Inactive	0
PF[20]..[28]	Reserved		
PF[29]	Permitted acceleration per 1 Amp of drive current Applicable only if UM=3 *and* PN[9]=1	100..60000000	10000000

Attributes:	**Type:**	Parameter, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Index range:	[1,11]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	No


Under most circumstances, this command is used only by the tuning environment.

See also:

PF[18],PF[19]: - MF

Application notes:

PF[1]: Motor inductance corrections

PF[2]: BEMF corrections
PF[3,4,5]: Cogging correction

PF[9],PF[12]: Current loop implementation

PF[16],PF[17]: Torque and position reference generation for open loop stepper

PF[29]: Closed loop stepper control

PL[N] - Peak Duration and Limit

Purpose:

- PL[1] defines the motor maximum peak current, in amperes.
- PL[2] defines the motor maximum peak duration, in seconds.

This parameter is used to protect the motor (or the drive) from over-current, and to protect the load from excessive torque. The motor current (torque) command is normally limited to its peak limit, as defined by PL[1]. After a short period of torque demand higher than C[1], the torque command limit is decreased to CL[1]. If the current command has been raised to PL[1] from zero, after the time specified for peak duration (PL[2], in seconds), the motor current command will be limited to CL[1]. The motor current command remains limited to CL[1] until enough time has passed for the average requested torque command to fall below 90% of CL[1].

The LC flag indicates that the current is limited to its continuous limit.

Torque limits PL[N] and CL[N] may be changed dynamically while the motor is on.

Attributes:		
	Type:	Parameter, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	PL[1]=0, PL[2]=3 (RS), Non-volatile
	Range:	PL[1]: [0…MC] PL[2]: [1…3]
	Index range:	[1, 2]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Modified

Notes:

- Always specify a PL[1] value that can be reached. Placing an unreachable PL[1] can cause the anti-windup mechanism not to work properly, causing large position overshoots when the drive saturates.
- Allowed peak current may be saturated at a level lower than the PL[1] value when the PWM frequency is increased with the XP[2] command. The actual peak saturated value in amperes can be retrieved with the WS[33] command. The option applies only to drives that have the ability to increase the PWM rate.
- If CL[1] $\geq$ PL[1], PL[1] will be the torque limit in effect at all times, and PL[2] will be ignored.
- The minimum current limit is MC/128. If PL[1] < MC/128, the PL[1] value will be accepted, but the actual current value will be limited to MC/128.
- PL[1] is used to determine the maximum and minimum limits for the PWM generator used for current reference in 50% and 100% PWM mode.

See also:
CL[N], LC, MC, TC, XP[N]

Application note:

Current limiting

PM - Profiler Mode

Purpose:
The PM command bypasses the motion profiler in UM=2 (speed mode).

With PM=1, the target speed is fed directly to the speed controller, without acceleration limits.

PM=0 is set by the tuner program to test the step response of the speed controller. Even with PM=0, the speed command acceleration and deceleration is limited by the SD parameter.

The values of PM are outlined in the following table:

PM Value	Description
1	Normal
0	For UM=2, bypass profiler.

Table 3-33: **PM Values**

Attributes:	**Type:**	Command/Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0
	Default value:	1, Non-volatile
	Range:	[0, 1]
	Unit modes:	UM=2
	Activation:	MO=1
	SimplIQ:	**Similar**

See also:
UM, AC, DC, SD, SF


Under normal circumstances this command is used by the tuning environment only.

PN[N] – Integer parameters

Purpose:

This parameter vector is used for device data, which is normally entered by the tuning environment. The user is encouraged to use the tuning environment instead of entering these parameters manually.

Parameter	Description	Units [range]	Default
PN[1]	Maximum amount of Hall sensor misses before exception emits. 0 or negative number cancels check.	Pure number, [0 … 8000]	4
PN[2]	0: Do not compensate cogging Non-zero: Compensate cogging	NA	0
PN[3]	0: Do not make speed compensations to current loop Non-zero: Make speed compensations to current loop	NA	0
PN[4]	Position reference counts per electrical revolution. In stepper motor terminology, PN[4]/4 is the "micro-stepping factor" – there are PN[4]/4 position counts in one step.	Pure number, [1…8192]	1024
PN[5]	Apply varying notch filter on reference for open loop stepper 0: Do not apply Otherwise: Apply	Pure number [0,1]	0
PN[6]	Re-find commutation angle each motor on 0: Do not apply Otherwise: Apply	Pure number [0,1]	0
PN[7]..PN[8]	Reserved		
PN[9]	1: Closed loop stepper mode (applicable only for UM=3)	Pure number [0,1]	0

Parameter	Description	Units [range]	Default
PN[10]	Index position for analog encoder, coming from the positive direction. Refer to the Application Note for details.	[0..65535]	0
PN[11]	Index position for analog encoder, coming from the negative direction. Refer to the Application Note for detail	[0..65535]	0

Attributes:

Type: Parameter, Integer
Source: Program, RS-232, CANopen
Restrictions: None
Index range: [1,11]
Unit modes: All
Activation: Immediate
SimplIQ: No

Note:

- For open loop stepping of rotary motors, XM[2]-XM[1] must be an integer multiple of PN[4], otherwise stepping angle jumps will appear on position modulo transitions.

Under most circumstances, this command is used only by the tuning environment.

Application notes:

PN[1]: Checking encoder against Hall sensors

PN[2]: Cogging correction

PN[3]: Speed correction

PN[4]: Stepper current and position reference generation

PN[5]: Torque and position reference generation for open loop stepper

PN[6]: Brushless DC commutation

PN[9]: Closed loop stepper control

PN[10], PN[11]: Index for analog encoder

PP[N] - Protocol Parameters

Purpose:
Programs all communication parameters. The PP[N] command has independent fields for the parameters of all supported communication methods. These parameters are tabulated in the tables that follow.

Parameter	Description	Range
PP[1]	Type of communication. PP[1] serves as "Enter Communication Parameters" for RS-232. PP[2] and PP[4] come into effect only when PP[1] is written. The response to PP[1]=x is not the same as the response to all other commands, because the communication type switches while processing the command.	1: RS-232 2: Reserved for compatibility with previous drives.
PP[2]	RS-232 baud rate. This parameter has no immediate effect.	5: 115,200; 4: 57,600 3: 38,400 2: 19,200 1: 9,600 0: 4,800
PP[3]	Spare.	
PP[4]	RS-232 parity.	0: None 1: Even 2: Odd

Table 3-34: **RS-232 Communication Parameters**

Parameter	Description	Range
PP[13]	CANopen device ID.	1 - 127
PP[14]	CAN baud rate.	0: 1,000,000 1: 500,000 2: 250,000 3: 125,000 4: 100,000 5: 50,000 6: 50,000 7: 50,000 8: 800,000
PP[15]	CAN group ID.	1 - 128

Table 3-35: **CAN Communication Parameters**

Attributes:	**Type:**	Parameter, Integer
	Source:	RS-232, CANopen
	Restrictions:	MO=0 for PP[1]
	Default values:	PP[1]=1, PP[2]=2, PP[13]=127, PP[14]=1, PP[15]=128 All others default to zero (RS), Non-volatile
	Range:	As shown in previous tables
	Index range:	[1…15]
	Unit modes:	All
	Activation:	RS-232 parameters activated by setting PP[1]. CANopen parameters activated upon restarting network through NMT service (refer to Elmo *CANopen Implementation Manual*).
	SimplIQ:	Only difference: 115.2 kBaud enabled

Notes:

- The number of R-232 stop bits is fixed to 1.
- The group ID number for CAN (PP[15]) defines the received message object ID. The response is transmitted by each node with its own ID (PP[13]). Setting PP[15]=128 allows the user to cancel the CAN group ID.
- Unused PP[N] parameters are reserved for compatibility with other Elmo drives.

PR - Relative Position

Purpose:
Specifies that the next software position command will be a PTP (point-to-point) motion and defines its target position (refer to the PA command). PR may be applied in any active motion mode; it is activated and applies changes to the PA setting only after the next BG is executed.

- *If PTP motion is already active,* PA will be increased by the PR value and become the new position target.
- *If PTP motion is not active* and is now activated by PR, PA will be set to the new value, which is equal to the software position command plus the PR value (PA=Position command + PR). This PA value will become the new target position.

Setting the PA value always resets the PR value to zero.

Attributes:		
	Type:	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=1
	Default value:	0, Volatile. Cleared automatically at MO=1
	Range:	See Notes below
	Unit modes:	UM=3, 4, 5
	Activation:	BG
	SimplIQ:	Similar

Notes:

- If a PA value was set but not activated by BG prior to activating this PR setting, the PA value will be discarded.
- The position target is calculated with modulo counting, resulting in the following range:
  [XM[1]…XM[2]] for UM=3, 5
  [YM[1]…YM[2]] for UM=4
  and restricted by VL[3] and VH[3].
  For example, in UM=5, if XM[1]=0 and XM[2]=1000, PR=2500 is similar to PR=500. PR is rejected if the results of PA+PR exceed the VL[3] and VH[3] limits.

See also:
PA, XM[N], YM[N]

Application note:
The Position Reference Generator

PS - Program Status

Purpose:
Returns the status of the user program. If a user program is running, PS returns the number of user program threads:

- 0 if the user program is halted.
- –1 if no user program thread is running.
- –2 if no user program is ready to run.

Attributes:	**Type:**	Status report, Integer
	Source:	RS-232, CANopen
	Restrictions:	None
	Unit modes:	All
	SimplIQ:	Similar

See also:
CC

SimplIQ Programming and Language Guide

PT - Position Time Command

Purpose:
Specifies that the next BG will start a PT (Position - Time) tabulated motion and defines the starting index in the QP[N] array.

In a PT motion, a new position reference value is picked from the QP[N] array once per MP[4] position controller sampling time. The motion is interpolated between the points specified by the QP[N] array. The MP[N] parameters specify which indices of the QP[N] array are used for the motion. After BG, the PT motion starts from position QP[PT].

A PT motion terminates when, in non-cyclical mode (MP[3]=0), the index of MP[2] is reached, or if the PT buffer underflows (CANopen only). When a PT motion terminates, the motor is brought to a complete stop using the SD deceleration.

When a PT motion is executing, PT reads the presently executing index of the QP[N] array.

Attributes:	**Type:**	Command/Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=1, PT motion executing, Advanced model only
	Default value:	None, Volatile
	Range:	[1…1024]
	Unit modes:	Position control (UM=3, 4, 5)
	Activation:	Next BG
	SimplIQ:	Similar

See also:
PV, MP[N], QP[N], QV[N], QT[N]

Application note:
The Position Reference Generator

PV - Position Velocity Time Command

Purpose:
Specifies that the next BG will start a PVT (Position - Velocity - Time) tabulated motion and defines the starting index for the QP[N], QV[N] and QT[N] arrays.

In a PVT motion, new position and speed reference values are picked from the QP[N] and QV[N] arrays at the time specified by the elements of the QT[N] array. The motion is interpolated between the entries of QP[N] and QV[N]. The MP[N] parameters specify which indices of QP[N], QV[N] and QT[N] are used for the motion.

After BG the PVT motion starts from the QP[PV] position.

A PVT motion terminates when, in non-cyclical mode (MP[3]=0), the index of MP[2] is reached, or if the PVT buffer underflows (CANopen only). When a PVT motion terminates, the motor is brought to a complete stop using the SD deceleration.

When a PVT motion is executing, PV reads the presently executing index of the PVT table.

Attributes:		
	Type:	Command/Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=1, PVT motion executing, Advanced model only
	Default value:	None, Volatile
	Range:	[1…64]
	Unit modes:	Position control (UM=3, 4, 5)
	Activation:	Next BG
	SimplIQ:	Similar

See also:
PT, MP[N], QP[N], QV[N], QT[N]

Application note:

The Position Reference Generator

PW[N] - PWM Signal Parameters

Purpose:

- PW[1] defines the offset value for PWM signals in fractions of the Duty cycle.
- PW[2] defines the dead band zone of the PWM signal in the fractions of the Duty cycle.

At times, the application requires the Duty cycle value to be different from the source of the PWM signals (i.e. offset). The PW[1] parameter is intended for this purpose.

The Actual Duty Cycle is *Estimated Duty Cycle – Offset*.

A very low Duty cycle values may cause a motor to "crawl" when a complete stop is desired.

The dead band zone parameter (PW[2]) forces the Actual Duty Cycle to zero when it is less than PW[2].

Attributes:	**Type:**	Parameter, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	PW[1]=0, PW[2]=0(RS), Non-volatile
	Range:	PW[1]: [-1.0 …1.0]
		PW[2]: [0.0 …1.0]
	Index range:	[1, 2]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

Note:

- A *SimpIIQ* drive may receive the PWM signals in two formats: 50% and 100% Duty cycle. For more details, please refer to the YA[] command.

See also:
AN[N], FR[N], RM, YA[N]

PX - Main Position

Purpose:
Reads the position of the main feedback. Upon power on, the main position is set to zero. The PX variable accumulates the main feedback pulses. PX can count cyclically (refer to the XM[N] command). When the motor is off, PX may be used to set a value for the position counter by typing PX=n. To program the position while the motor is on, refer to the HM[N] command.

Attributes:	**Type:**	Parameter/Status report, Integer
	Source:	Program, RS-232, CANopen
	Reset value:	0, Volatile
	Restrictions:	MO=0
	Range:	See Notes below
	Unit modes:	All
	SimplIQ:	Similar

Notes:

- PX is limited to the position counter range.
- If PX exceeds the modulo range, it will be automatically set to a value *within* that range.
- PX counts in the direction defined by CA[16]. If CA[16] is modified, PX will not change, but position counting will be continued in the other direction. To ensure proper commutation, CA[16] must be modified in conjunction with the motor direction (CA[25]).

See also:
HM[N], XM[N], CA[N]

PY - Auxiliary Position

Purpose:
Reads the position of the auxiliary feedback. Upon power on, the auxiliary position is set to zero. The variable PY accumulates the auxiliary feedback pulses. PY can count cyclically (refer to the YM[N] command). When the motor is off, PY may be used to set a value for the auxiliary position counter by typing PY=n. To program PY while the motor is on (MO=1), refer to the HY[N] command.

Attributes:	**Type:**	Parameter/Status report, Integer
	Source:	Program, RS-232, CANopen
	Reset value:	0, Volatile
	Restrictions:	MO=0
	Range:	See Notes below
	Unit modes:	All
	SimplIQ:	Similar

Notes:

- PY is limited to the position counter range.
- If PY exceeds the modulo range, it will be automatically set to a value *within* that range.
- PY counts in the direction defined by YA[5]. If YA[5] is modified, PY will not change, but position counting will be continued in the other direction. In dual feedback mode (UM=4), the motor direction must be changed accordingly.

See also:
HY[N], YM[N], YA[N], CA[N]

QP[N], QT[N], QV[N] - Position, Time, Velocity

Purpose:
Stores data for the PT and PVT motion modes. The QP[N], QV[N] and QT[N] arrays define the position (QP[N]) and speed (QV[N]) at any single time instance (QT[N]).

With CANopen communication, the QP[N], QV[N] and QT[N] arrays can be programmed at high speed using specially designed communication objects.

Attributes:	**Type:**	Command/Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default value:	0, Volatile
	Range:	QP[N]: [-2^{31}…2^{31} – 1]
		QT[N]: [1…255], in msec
		QV[N]: [-2^{31}…2^{31} – 1]
	Index range:	QP[N]: [1…1024]
		QT[N]: [1…64]
		QV[N]: [1…64]
	Unit modes:	[3…5]
	Activation:	Immediate
	SimplIQ:	Similar

The values of QP[N] and QV[N] are unlimited, to enable their use as general-purpose storage when PVT/PT is not used. However, for PVT/PT, the QP[N] and QV[N] values should be confined to the position and velocity ranges.

See also:
PT, PV, MP[N]

Application note:
The Position Reference Generator

RC - Define Recorded Variables

Purpose:
Defines which signals are to be recorded.

The drive can record a range of signals for performance verification and debugging. The first step of the recording process is the definition of the recorded variable by assigning a value to RC, a bit field. Each "on" bit in the binary representation of RC defines a signal to be recorded. The host can map many optional variables to any bit of RC from bit 0 to bit 15.

A valid RC defines at least one recorded variable. Up to eight variables can be selected.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	Recorder inactive (RR=0 or RR=-1)
	Default value:	0 (RS), Non-volatile
	Range:	See previous description
	Unit modes:	All
	Activation:	Next initiation of recorder (RR)
	SimplIQ:	Similar – see note


Under most circumstances, this command is used only by the Composer program and the tuning environment.


Notes:

- If the drive has stored previously-recorded data, setting RC will invalidate this data. Invalidated data cannot be retrieved.
- The total number of data points that may be recorded is fixed. Therefore, the number of points per signal depends on the number of signals recorded simultaneously: the more signals recorded, the fewer points available for each signal.
- SimplIQ difference: with AR[1]=1, this command refers to the motion processor.

See also:
RG, RL, RP[N], RR, BH,AR[N]

Application note:
The Recorder

RG - Recorder Gap

Purpose:
Defines the frequency per sampling times that the recorder is activated.

Because the recorder has a limited storage capacity, if it operates at the sampling time of the drive, the recorder will operate for a very short time. For longer recording times, the time interval between consecutive data recordings must be increased. The RG parameter trades recording resolution against recording time. With RG=1, the sampling time of the recorder is WS[29]. Note that the recorder sampling time depends on the TS value and the specific unit mode (UM).

Attributes:		
	Type:	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	Recorder inactive (RR=0 or RR=-1)
	Default value:	1 (RS), Non-volatile
	Range:	[1…4096]
	Unit modes:	All
	Activation:	Next initiation of the recorder (RR)
	SimplIQ:	Similar (see note)


Under most circumstances, this command is used only by the Composer program and the tuning environment.

- SimplIQ difference: with AR[1]=1, this command refers to the motion processor.
- If the drive has stored a previously recorded data vector, setting RG will invalidate this data. Invalidated data cannot be retrieved.

See also:
RC, RL, RP[N], RR, BH, TT[N], WS[29], UM

Application Note:
The Recorder

RL - Record Length

Purpose:
Specifies the length of the recorded data, as follows:

Number of Simultaneously Recorded Signals	Maximum Record Length
1	4096
2	2048
3	1365
4	1024

Table 3-36: **Record Length of Simultaneously Recorded Signals**

RL can specify that the signal records will be shorter than the maximum. If RL is set to a value higher than the maximum record length (for example, RC defines three recorded signals, but RL=2048), the recorder will still work. The length of the records, however, will be shorter than RL. The actual size of the recorded data is returned by WI[21].

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	Recorder inactive (RR=0 or RR=-1)
	Default value:	256 (RS), Volatile
	Range:	[1…4096]
	Unit modes:	All
	Activation:	Next initiation of the recorder (RR).
	SimplIQ:	Similar (see note)


Under most circumstances, this command is used only by the Composer program and the tuning environment.


Notes:

- SimplIQ difference: with AR[1]=1, this command refers the motion processor.
- On AR[1]=1 (Motion processor recording), the maximum record length differs from the specification in the table above.
- If the drive has stored a previously recorded data vector, setting RL will invalidate this data. Invalidated data cannot be retrieved.

See also:
RC, RG, RP[N], RR, BH

Application note:
The Recorder

RM - Reference Mode

Purpose:
Specifies the use of an external reference signal. In all unit modes, the SimpIIQ drive sums the reference command to the drive from two sources:

- A software command, generated internally either by a communicated command or by a user program
- An "auxiliary" reference, may be the sum of analog input #1 and the external signals that are derived by the auxiliary encoder (possibly using the ECAM table) or PWM signals.
- In position mode (UM=4, UM=5), the analog input is not used as a reference.
- In stepper mode (UM=3), an auxiliary reference sets the torque value to the stepper generator.

The RM values are as follows:

RM Value	Meaning
0	The reference command is generated internally, either by the interpreter command or by the user program.
1	The reference command is summed from the internal software reference command and the auxiliary reference command.

Table 3-37: **RM Values**

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	Defined in following notes
	Default value:	0 (RS), Non-volatile
	Range:	[0, 1]
	Unit modes:	All
	Activation:	MO=1
	SimplIQ:	Similar

Notes:

- If you are not using the auxiliary referencing option, set RM=0 to prevent A/D noise from falsely driving the drive.
- In dual loop mode (UM=4), switching to auxiliary reference mode (RM=1) is not allowed.
- In position mode (UM=5), RM may be changed only when the software reference is stationary (MS ≤ 1). In speed mode (UM=2) RM may be changed at any time.

See also:
UM, FR[N], AG[N], AS[N], PW[N]

Application note:
Unit Modes

RP[N] - Recorder Parameters

Purpose:
Enables the complete specification of how the recorder is triggered and how the recorded data is transferred to the host.

Trigger definitions:
The recorder is started by a trigger event, which may be one of the following:

- *Immediate:*
  The recorder starts immediately after the recording request has been issued.
- *Triggered by an analog signal:*
  The recorder starts upon one of the following events:
  - Positive slope: The signal crosses a prescribed level with a positive slope.
  - Negative slope: The signal crosses a prescribed level with a negative slope.
  - Window: The signal exits a window of two prescribed signal levels.
  - Digital inputs: Digital inputs are switched to their active logic state as defined by the IL[N] command.
- *Motion begins:*
  A BG command, or the activation of a hardware BG command (refer to the IL[N] command).

Trigger delay:
The trigger defines when the recorder is to start. The recorder can be programmed to start before the trigger event, so that the trigger event can be caught "in the middle of the action." This is possible because the recorder starts to record at the instant it is launched by the RR command, so that when the trigger event occurs, the pre-trigger information is already recorded. The trigger parameters are listed in the following table:

RP[N]	Range	Definition
RP[0]	[0…1]	0: Time quantum is 4*TS 1: Time quantum is TS
RP[1]: Trigger variable	[1…65,536]	Defined similarly to RC, but only 1 bit may be non-zero. The trigger variable does not need to be one of the recorded variables.
RP[2]: Pre-trigger storage in percent	[0…100]	The percentage of the recorded signal taken before the trigger event.
RP[3]: Trigger type	[0…5]	0: Immediate 1: BG 2: Positive slope 3: Negative slope 4: Window 5: Digital inputs

RP[N]	Range	Definition
RP[4]: Level 1	Unlimited	Level for positive slope trigger, or high side for window trigger.
RP[5]: Level 2	Unlimited	Level for negative slope trigger, or low side for window trigger.
RP[6]: Polarity	[0…63]	Defines the polarity for the digital input trigger of the recorder. 1: Positive polarity.
RP[7]: Digital input mask	[0…63]	Defines which digital inputs trigger the recorder.

Table 3-38: **Trigger-related RP[N] Parameters**

The following parameters enable the BH command to fetch only the required part of the recorder results:

RP[N]	Range	Definition
RP[8]: Index low	[0…1024]	Lower buffer index for recorded data transmission.
RP[9]: Index high	[0…1024]	Higher buffer index for recorded data transmission. When RP[9]=RP[8]=0, all of the buffer is transmitted.

Table 3-39: **RP[N] Parameters Related to the BH command**

Attributes:

Type:	Parameter, Integer
Source:	Program, RS-232, CANopen
Restrictions:	None
Default values:	RP[1] is set by the UM command. The other RP[N] values are defaulted to 0 at power on
Range:	Defined in previous tables
Index range:	[1…9]
Unit modes:	All
Activation:	Trigger parameters: RR=3 Upload parameters: Immediate

If the drive has stored a previously recorded data vector, setting RP[N] (with N other than 8 or 9) will invalidate this data. Invalidated data cannot be retrieved.

See also:
RC, RG, RL, RR, BH

Application note:
The Recorder

RR - Activate Recorder / Get Recorder Status

Purpose:
Launches the recorder, kills an on-going recording process or retrieves the recorder status. The RR command has the following options:

RR Value	Meaning
0	Kill the recorder (do nothing if the recorder is not active).
1	Start recording at the next BG command.
2	Start recording immediately.
3	Arm the recorder with the trigger settings of the RP[N] parameters.

Table 3-40: **RR Command Options**

As a status report, RR may return the following values:

RR Report	Meaning
-1	There is no valid data in the recorder.
0	The recorder action is complete and it is loaded with valid data.
1 - 3	Waiting for the completion of RR=1, RR=2 or RR=3, respectively. The report value 3 does not indicate if the recorder is already recording or just waiting for a trigger. If this differentiation is required, use the SR command.

Table 3-41: **RR Reports**

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default value:	-1, Non-volatile
	Range:	[0…3]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar (see note)


Under most circumstances, this command is used only by the Composer program and the tuning environment.


Note:

- SimplIQ difference: with AR[1]=1, this command refers to the motion processor.

See also:
BH, RP[N], RC, RG, RL, RR

Application note:
The Recorder

RS - Soft Reset

Purpose:
Initializes the drive parameters to their factory default, and resets all volatile variables to their power-on default.

Attributes:	**Type:**	Command, No value
	Source:	RS-232, CANopen
	Restrictions:	MO=0, Program not running
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar – see note

Notes:

- RS does not change the communication settings; therefore, after executing RS, it is still possible to communicate with the drive. The communication parameters, however, are reset. For example, if the baud rate is 9600, setting PP[1]=1 immediately after RS will switch the baud rate to the default of 19,200.
- The RS command disables the communication routines for a few milliseconds. If an RS is executed by an RS-232 command, a CAN message may be lost in the execution interval.
- The RS command modifies only the RAM contents; it does not affect the flash memory. Use the SV command to make the effect of RS permanent.
- After an RS command, the current limits are set to zero so it is impossible to start the motor immediately. After an RS, it is recommended to go through the steps of the Composer Wizard before attempting to work with the motor.
- On major software updates, the database may become incompatible. In that case, the drive internally resets to factory defaults.
- RS does may leave the motion tables unsynchronized to the motion processor – refer to the SI command.

See also:
LD, SV, CD, SI

RV[N] - Recorded Variables

Purpose:
Maps recorded variables to the recorder for the RC command.

The RC command selects which signals will be recorded. Historically, there were only 16 recordable signals which the user could select. The RC command selected which signals to record out of the available 16.

SimplIQ supports far more than 16 recordable signals – the list of recordable signals may vary with the software version and is stored internally in the drive.

RV[N] =X redefines to which signals the RC command refers:

By setting RV[N]=M, bit N-1 of RC is assigned the #M variable in the signals list.

Attributes:	**Type:**	Parameter, Integer, Bit-field
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	RV[1]=1, RV[2]=2 . . . RV[16]=16 (RS), Volatile
	Range:	According to static table variable index.
	Index range:	[1…16]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Modified (see note)


Under most circumstances, this command is used only by the tuning environment.

- SimplIQ difference: with AR[1]=1, this command refers to the motion processor.

See also:
RC, BH, RR,AR[N]

Application note:
The Recorder

SD - Stop Deceleration

Purpose:
Defines the deceleration in counts/second^2 used to stop motion in case of emergency. In addition, SD defines the acceleration limit for the combination of software and external reference commands.

Position-controlled motions cannot be stopped abruptly, because:

- The discontinuity in the reference speed may produce position errors in excess of ER[3]. This cuts the motor drive to freewheeling, which may be a safety problem with high inertia or fast-moving loads.
- If the maximum allowed acceleration value (GS[9]) is not correctly set, generating a large position error can destabilize the position controller.

The SD parameter must be defined as the largest deceleration in which the motor can accelerate/decelerate to the load.

The SD acceleration is applied in the following cases:

- An ST command or hardware stop command (refer to the IL[N] command).
- Data underflow in a PT or PVT motion.
- Detection of a limit switch or an abort switch.

Attributes:		
	Type:	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0
	Default value:	1,000,000,000 (RS), Non-volatile
	Range:	Stop deceleration range
	Unit modes:	UM=2, 3, 4, 5
	Activation:	MO=1
	SimplIQ:	Modified (range extended)

See also:
AC, DC, ST

Application notes:
The Position Reference Generator

Unit mode 2: Speed control

SI - Synchronize Motion Processor Tables

Purpose:
Transfers the motor description tables to the motion processor.

The motion tables describe some fine motor modeling aspects that help to improve control quality. Among them are cogging compensation tables, as well as winding shape compensation tables.

Since this tabular data must be available to the motion processor on real-time, it must be downloaded as a block to the motion processor before any motion starts.

The SI command downloads the entire table block, as programmed by the TV command, to the motion processor, and stores it in the flash memory there.

Attributes:	**Type:**	Command, with integer parameter
	Source:	RS-232, CANopen
	Restrictions:	MO=0, User program not running
	Range	1: Activate
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	New command


Under most circumstances, this command is used only by the tuning environment.


Notes:

- The SI command may take a few hundreds of milliseconds to execute, during which the communication drivers are disabled. If an SI command is executed by an RS-232 command, a CAN message may be lost in the interim, and vice versa.
- SI synchronizes the contents of the TV[N] vector with the motion processor. The stored TV[N] commands are the last programmed set, not the copy stored in the flash memory.
- This command operates on the motion processor's flash memory. Note that the flash memory is limited by the number of times data may be saved to it (about 10000 saves). Therefore, never use SI in a user program.
- SI is only relevant to implementations that have a separate motion processor**.** For single processor implementations SI will not return an error – it will simply do nothing.

See also:
CD, TV[N], LD

Application notes:
Storing non-volatile application data

The motion processor

SF - Smooth Factor

Purpose:
Defines the motion smoothing factor for PTP and jogging motions. Smoothing means that the motion speed profile has no "sharp corners". The price for smoothing is that the total time required for completing the motion increases. For SF>0, the acceleration to the required speed is not set immediately to its final value but takes SF milliseconds to build. The total time required to complete the motion is increased by SF milliseconds. The maximum SF for the actual sampling time, in milliseconds, is given by WI[22].

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0
	Default values:	0 (RS), Non-volatile
	Range:	[0…100]; See Notes below
	Unit modes:	UM=2, 3, 4, 5
	Activation:	BG
	SimplIQ:	Similar

Notes:

- WI[22] reports the maximum permitted value for SF. Setting SF greater than WI[22] is not reported as an error, but the smoothing factor that is applied may be less than the actual setting. WI[22] depends on the sampling time (TS command).
- The smoothing action applies for PTP and jogging motions only. SF does not affect tabulated motions such as PT and PVT.
- If a PT/PVT motion command is given while a smoothed motion is on, a jump in the software reference may occur due to the removal of the smoothing filter. The stop manager will limit the rate at which the position command catches up with the software command that has jumped.
- If a PTP or jog motion command is given while PT/PVT motion is on, the smoothing effect will gradually build up, until complete smoothing is reached after SF milliseconds.
- SF does not affect the external velocity or position reference.

See also:
AC, DC, JV, SP

Application note:
The Position Reference Generator

SN - Serial Number

Purpose:
Returns the contents of CANopen object 0x1018 (LSS protocol) as an integer. The LSS protocol defines the behavior of the CAN node for ID setting and baud rate changing. Object 0x1018 includes the identification of the specific CAN node in such a way that the identification is totally unique.

The identification number contains four entries represented in SN[N] as follows:

- SN[1] returns the vendor ID.
- SN[2] returns the product code.
- SN[3] returns the revision number.
- SN[4] returns the serial number.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default value:	0, Volatile
	Range:	32 bits
	Index range:	[1…4]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

SP - Speed for PTP Mode

Purpose:
Sets the maximum speed for PTP (point-to-point) motion. At the start of motion, the speed of SP is reached with the acceleration of AC. Then, a constant speed of SP is maintained until the deceleration to final stop begins with the DC acceleration.

The speed of SP counts/second is achieved only if the motion is long enough, if AC and DC are large enough, and if SF is small enough. Otherwise, the speed limit of SP remains inactive.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	|VL[2]| ≥ SP and VH[2] ≥ SP
	Default value:	25,000 (RS), Non-volatile
	Range:	Velocity range
	Unit modes:	UM=3, 4, 5
	Activation:	BG
	SimplIQ:	Similar (see note)

Notes:

- On AR[2]=1, this command routes directly to the motion processor.

See also:
HL[N], LL[N], AC, DC, SF, PA, PR

Application note:
The Position Reference Generator

SR - Status Register

Purpose:
Returns a bit-field, reporting the status of the system in a concise format. Most of the information in SR may be recovered using other commands. The primary purpose of the SR command is to enable a remote host — such as the Composer program — to get a snapshot of the system status without overloading the communications system.

Reported Status	Bits
Drive read: 0: Conditions OK 1: Problem, as reported by bits 1 - 3	0
Servo drive status indication details: refer to the following table	1 - 3
Motor on (MO)	4
Reference mode (RM)	5
Motor failure latched (see MF for details)	6
Unit mode (UM)	7 - 9
Gain scheduling on	10
Either Main or Auxiliary Homing being processed	11
Program running	12
Current limit on (LC)	13
Motion status reflection (MS)	14 - 15
Recorder status: 0: Recorder inactive, no valid recorded data 1: Recorder waiting for a trigger event 2: Recorder finished; valid data ready for use 3: Recording now	16 - 17
Not used	18 - 23
Digital Hall sensors A, B and C[5]	24 - 26
CPU status: 0: CPU OK 1: Stack overflow or CPU exception	27
Stopped by a limit – RLS, FLS, Stop switch – or by a VH[3]/VL[3] position command limit	28
Error in user program	29
Unused	30 - 31

Table 3-42: **Status Register Bits**

[5] The digital Hall sensor readings in SR are corrected according to CA[1…6].

0x8	0x4	0x2	Meaning
0	0	0	OK.
0	0	1	Under-voltage: The power supply is shut off or it has too high an impedance.
0	1	0	Over-voltage: The power supply voltage is too large, or the servo drive did not succeed in absorbing the kinetic energy while braking a load. A shunt resistor may be needed.
1	0	1	Short circuit: The motor or its wiring may be defective.
1	1	0	Temperature: The drive is overheating.

Table 3-43: **Servo Drive Status Indications**

Attributes:	**Type:**	Status report, Integer, Bit-field
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Unit modes:	All
	SimplIQ:	Similar

See also:
MF, RR, PM, UM, RM, IP, MS, MO, LC, HM[N], HY[N]

ST - Stop Motion

Purpose:
Stops the software motion. The software commands decelerate to a complete stop using the SD deceleration. ST does not affect the external position reference and has no effect for the following conditions: MO=0 and UM=1.

Attributes:	**Type:**	Command, No value
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Unit modes:	UM=2, 3, 4, 5
	Activation:	Immediate
	SimplIQ:	Similar

See also:
BG, RM

Application notes:
The Position Reference Generator

The Speed Reference Generator

SV - Save Parameters to Flash

Purpose:
Saves the entire set of non-volatile variables from the RAM to the flash memory. Before saving, the parameter integrity is tested. If the test fails, the SV command exits with an error and the flash contents remain as is. The CD command details the reason for the failure.

Attributes:	**Type:**	Command, No value
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0, User program not running
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Modified

Notes:

- The SV command may take a few hundred milliseconds to execute, during which the communication drivers are disabled. If an SV command is executed by an RS-232 command, a CAN message may be lost in the interim, and vice versa.
- Note that the flash memory is limited by the number of times data may be saved to it (about 100,000 saves). Therefore, be very careful when defining the use of SV in a user program.
- The SV command stores the motor mode correction tables (TV[N]), but does not transfer them to the motion processor. In order to complete the motion table transfer to the motion processor use the SI command.

See also:
CD, SI, LD,TV[N]

Application notes:
Storing non-volatile application data

The motion processor

TC - Torque Command

Purpose:
Sets the torque (motor current) command, in amperes, for the torque-control software-reference modes (UM=1). TC commands are accepted in the range permitted by the present torque command limits (refer to the PL[N] and CL[N] commands).

If TC is set to greater than CL[1], after a few seconds the current limit of the servo drive will drop to CL[1].

TC defines the reference value IQ (the ID command is always zero). When RM=1, TC is summed with the external analog commands.

Attributes:	**Type:**	Command/Parameter, Real
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=1, UM=1 or UM=3
	Default value:	0, Volatile. Cleared automatically at MO=1
	Range:	Torque limits
	Unit modes:	UM=1, 3
	Activation:	Immediate
	SimplIQ:	Modified (does not apply to UM=3)

See also:
MO, UM, IQ, ID, CL[N], PL[N], MC, HT[N]

Application note:

Unit Modes

TI[N] – Temperature indications array

Purpose:
Reports the drive temperature measurement:
- TI[1] – reports the drive temperature in Celsius.
- TI[2] – Reports the temperature at which the drive shuts itself with over-temperature protection.

Drive temperature is measured every ~3 mSec.

It is useful to analyze the drive temperature characteristics during the application development stages.

This feature can also serve as maintenance procedure during the machine lifetime.

The temperature is measured for over-temperature protection. The temperature sensor works well in the 25 °C to 130 °C range. Below 25^{c} measurements may be out of range and displayed as approximately 25^{c}.

Methods to obtain the temperature:
- Terminal command– query TI[1] by the host application.
- Record the temperature characteristic as a function of time by using the Composer recorder. Map the "Temperature" record variable from the record-mapping list.
- Using the drive's user program to sense a certain temperature, perform some of the following actions:
  - Stop the machine under control.
  - Reduce the current limits or change the motion profile.
  - Change a digital output level.
  - Transmit a CAN message to PLC or machine host.

Notes:
- When the temperature reaches the critical level, the drive will automatically shut down the power stage with a motor failure exception (MF=0xD000).
- If the drive does not support this feature, the TI[1] command returns –55.

Attributes:

Type:	Status report, Integer
Source:	Program, RS-232, CANopen
Restrictions:	None
Unit modes:	All
SimplIQ:	Similar

See also:
MF, RC

TM - System Time

Purpose:
Reads and writes the system time, in microseconds. *SimplIQ* drives have a 32-bit microsecond counter. In the absence of CAN SYNC and TSTAMP signals, the microsecond counter runs freely, completing a cycle approximately once every 71.5 minutes. The CAN SYNC and TSTAMP sequence synchronize this counter to the microsecond counter of the network master (refer to the *SimplIQ for Steppers CAN Implementation Manual*).

The BT command uses the TM variable to coordinate the start of motion. The tdif( ) function uses TM to calculate time differences.

Notes:

- When CAN communication is used, setting or updating the system counter by the user may upset the synchronization with the CAN network master.
- The tick system function also reads the system time of the *SimplIQ*, drive, but its value differs from the TM command value.

Attributes:		
	Type:	Command/Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default value:	None, Volatile
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

Examples:

QP[1023]=TM	Read TM into a software variable (here, QP[1023] is used as general storage, not as motion reference).
tdif(QP[1023])	Return the time elapsed since TM was sampled into QP[1023].
TM=0	Reset system time to zero.

The time returned by this command may jump due to a CAN master update (refer to the Elmo *CANopen Implementation Guide*).

See also:
BT

SimplIQ Programming and Language Guide

TR - Target Radius

Purpose:
Provides the criterion for deciding that a motion is complete and the motor is stabilized in place with the required accuracy, defined in terms of target radius and target time. The target radius is the maximum positioning error allowed for static stabilization (not to be confused with the ER[N] parameters, which represent the dynamic stabilization error that is considered a fault). The target time is the minimum time that the absolute value of the error must be within the target radius in order to determine that the motor has stabilized in its place after PTP (point-to-point) motion.

- TR[1] defines the target radius in counts.
- TR[2] defines the target time in milliseconds.

When a target position range is within the target radius (TR[1] command) for at least the target time (TR[2] command), the MS status is set to 0.

The TR[1] value must be within the limits of the modulo of the position feedback: (XM[1]…XM[2]) for UM=5 and (YM[1]…YM[2]) for UM=4.

Attributes:		
	Type:	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	TR[1]=100, TR[2]=20 (RS), Non-volatile
	Range:	TR[1]: [0…32,000], TR[2]: [0…100]
	Unit modes:	UM=4, 5
	Activation:	Immediate
	SimplIQ:	Similar

See also:

MS

Application note:
Idle Mode and Motion Status

TT[N] – Motion Processor Sampling Time

Purpose:
This command sets the sampling time of the motion controller.

TT[1]: Sampling time of the motion processor, µsec.

TT[1] is the sampling time of the current loop. The sampling time of the velocity controller and of the position controller is two times TS. For example, if TT[1]=40, the torque/commutation controller runs once every 40 microseconds, the speed controller and the position controller executes every 80 microseconds.

For all unit modes, WS[28] and WS[55] give the actual sampling time of the speed and of the position controllers[6].

Notes:

- When TT[1] is increased, the current loop gains must be revised in order to prevent instability of the current loop, which may damage the drive and/or the motor.
- For > 1KW drives, the selection of TT[1] is a compromise between high servo performance and switching losses in the drive. Lower TT[1] implies increase in the available control bandwidth, but also implies increased switching rates of the power stage, thus increased heating.

Attributes:		
	Type:	Parameter, Integer
	Source:	RS-232, CANopen
	Restrictions:	Motor off
	Default values:	40 µsec
	Index range:	[1]
	Range:	[30..60]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	New command

Note: This command takes a few milliseconds to perform.

See also:
WS[N],WI[N]

Under most circumstances, this command is used only by the tuning environment.

Application note:

The motion processor

[6] There are two readouts for the same thing since historically in SimpleIQ the sampling times for speed and position were different.

TV[N] – Motion Processor Tables

Purpose:
This command enters data to the motor model correction tables.
The data includes, among others:

- Cogging correction tables
- Winding back EMF tables

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	Nominal (sinusoidal winding, no cogging correction), NonVolatile
	Index range:	[0…4095]
	Unit modes:	All
	Activation:	By SB command
	SimplIQ:	New command

See also:
SI, PF[N]


Under most circumstances, this command is used only by the tuning environment.


Entering a value to the TV[N] command may invalidate the data tables synchronization between the SimplIQ processor and the motion processor. After this, "Motor On" will be impossible until the use of the SI command.

Application notes:

Cogging correction

Winding shape corrections

TW[N] – Wizard Command

Purpose:
Contains parameters for internal use only. For example, the command is used for auto-tuning or debugging.

Attributes:	**Type:**	Report, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Range:	Not applicable
	Index range:	[1…32]
	Unit modes:	All
	SimplIQ:	Modified


Under most circumstances, this command is used only by the tuning environment.


Notes:

- This command works only after entering TP[6]=1 in order to prevent accidental activation.
- Not all the TW commands are documented here – only those that are more likely to be used.

Parameter	Description	Units [range]	Default
TW[63]	Sets a value for the commutation counters and tells the amplifier that commutation is known, avoiding the commutation search the next time the motor is turned on.	0…CA[18]-1 (not checked)	None

Application note:

TW[63]: Setting the commutation counter manually

UF[N] – User Float Array

Purpose:
Provides an array of 24 floating numbers for general-purpose use.

Attributes:	**Type:**	Parameter, Float
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	0 (RS), Non-volatile
	Range:	[-1e20…1e20]
	Index range:	[1…24]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

Typical Applications:
General look-up tables of real numbers and parameters for machine task definition.

Example:
See the UI[N] example.

See also:
UI[N]

UI[N] – User Integer

Purpose:
Provides an array of 24 integer numbers for general-purpose use.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default values:	0 (RS), Non-volatile
	Range:	$[(-2^{30}+1)\ldots(2^{30}-1)]$
	Index range:	[1…24]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

Typical applications:
General look-up tables of real numbers and parameters for machine task definition.

Example:
```
MO=0;
int MY_VAR ;
MY_VAR=10 ;
UI[2]=MY_VAR ;
for UI[1]=1:10
UF[UI[1]]=UI[2]*SP/UI[4];

end
```
In this example, the UF[N] array is dynamically indexed and valued by elements of the UI array.

See also:
UF[N]

UM - Unit Mode

Purpose:
Defines the motion controller drive configuration, as follows:

UM Value	Description (Related Commands)
0	*Voltage control mode* In this mode, the motor voltage is set directly by the VC software command. This mode is useful only for very special cases, for motors with very high resistance and very short mechanical time constants. Using this mode without extreme care may damage the motor.
1	*Torque control mode* In this mode, the motor current command is set directly by the TC software command or by an analog reference signal. This mode is useful for torque or force control when the servo drive is used only as an inner device within an external feedback loop.
2	*Speed control mode* In this mode, the motor speed command is set directly by the JV software jogging command or by an analog reference signal.
3	*Micro-stepper mode* In this mode, no commutation is made. The user controls the electrical field angle using the position commands and the motor current (holding torque) by the HT[N] command or by an analog signal. All position mode commands (PA, JV or tabulate motion) are applicable. This mode cannot be used with DC motors.
4	*Dual feedback position control* In this mode, the position controller stabilizes the position of the auxiliary feedback input. The position controller issues a motor speed command to an inner speed control loop. The inner speed control loop derives its speed feedback from the main feedback input. In this case, auxiliary reference mode (RM=1) is not allowed. The position command is generated similarly to UM=5, except that the auxiliary feedback is not available for reference generation.
5	*Single loop position control* In this mode, the position controller stabilizes the position of the main feedback input. The position command is summed by a software command — point-point (PA), jogging (JV) or tabulated motion (PT, PVT) — and from an external command. The external command is derived from the auxiliary feedback reference (FR[N], ET[N], EM[N]).

Table 3-44: **UM Values**

Notes:

- The unit mode is reflected in the SR report.
- SimplIQ for Steppers has UM=0, which SimplIQ did not have.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0,
	Default value:	3 (RS), Non-volatile
	Range:	[0…5]
	Activation:	Immediate
	SimplIQ:	Modified (added UM=0)

See also:
RM, SR, TT[N], WI[N]

Application note:
Unit Modes

VC[N]- Voltage Command

Purpose:
Defines the voltage output of the Q and the D controllers, over-riding the control functions.

VC[1] sets VQ

VC[2] sets VD

The units for VQ and VD are fractions of the available bus voltage.

Attributes:	**Type:**	Parameter, float
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=1
	Default value:	0 -volatile
	Range:	[0…0.95]
	Unit mode	0
	Activation:	Immediate
	SimplIQ:	New command

Note:

- VQ and VD do not compensate on line voltage changes.

See also:
UM

Application notes:
The current controller

Current output transformation

VE - Velocity Error

Purpose:
Reports the velocity tracking error:
VE = DV[2] – VX

If the absolute value of VE exceeds ER[2], motion is aborted and a motion fault code MF=128 (0x80) is set.

If MO=0, or if the speed controller is not used (UM=1, 3), VE returns 0.

Attributes:	**Type:**	Status report, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Unit modes:	UM=2, 4, 5
	SimplIQ:	Similar

Application note:

The Speed Controller

VH[N], VL[N] - High and Low Reference Limit

Purpose:
Define the drive's minimum and maximum speed and position reference limits. Software commands beyond these values are not accepted, and are truncated to VL[N] and VH[N].

- The reference to the speed controller is limited to the [VL[2]…VH[2]] range.
- The reference to the position controller is limited to the [VL[3]…VH[3]] range.

Attributes:	**Type:**	Command/Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0 VH[N] > VL[N] for all entries
	Default values:	VH[2]=15,000,000, VH[3]= 999,999,990 VL[2]=-15,000,000, VL[3]=-999,999,990 (RS), Non-volatile
	Range:	0 < VH[2] < max(Velocity range) min(Velocity range) < VL[2] < 0 VL[2]: Velocity range $-2^{31} \leq VH[3] \leq 2^{31} - 1$ $-2^{31} \leq VL[3] \leq 2^{31} - 1$
	Unit modes:	VH[2], VL[2]: UM=2, 3, 4, 5 VH[3], VL[3]: UM=3, 4, 5
	Activation:	Immediate
	SimplIQ:	Similar

Notes:

- In position modes (UM=4, 5) motor movement is enabled in both directions *within* the defined position reference range. If feedback has been extended beyond those limits, the motor can be enabled by the user (MO=1) but the motion can only be in the direction *towards* the reference limit range.
- VH[2] is used to determine the maximum and minimum allowed speed for the PWM reference generator in 50% and 100% PWM modes.

See also:
XM[N], LL[N], HL[N], RM

VR - Firmware Version

Purpose:
Reports the version of the firmware as a string, which includes:

- The product name.
- The software version.
- The software release date.

This command is intended for use only by RS-232 communication, or CAN OS.

Attributes:	**Type:**	Status report, String
	Source:	RS-232
	Restrictions:	None
	Unit modes:	All
	SimplIQ:	Similar

Examples:
VR
Bell 1.01.02.03 1Jul2007
where:

- The product name is Bell.
- The software version is 1.01.02.03.
- The software release date is 1 July 2007.

VX, VY - Velocity of Main and Auxiliary Feedback

Purpose:

- The VX status report returns the speed of the main feedback, in counts/second.
- The VY status report returns the speed of the auxiliary feedback, in counts/second.

The VX and VY signals are calculated using the time difference measured between consecutive feedback pulses.

Attributes:	**Type:**	Status report, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Unit modes:	All
	SimplIQ:	Similar

WI[N] - Miscellaneous Reports, Integer

Purpose:
Reports integer constants and variables of the system usually used by the Composer program rather than directly by the users.

Index	Description
WI[1]	Minimum sampling time for torque control mode.
WI[2]	Reserved.
WI[3]	Reserved.
WI[4]	Maximum sampling time for torque control mode.
WI[5]	Reserved.
WI[6]	Reserved.
WI[7]	Available CPU time for background tasks, remaining after real-time control algorithm performance.
WI[8]	First amplitude for the auto-phasing process.
WI[9]	Second amplitude for the auto-phasing process.
WI[10]	First oscillation phase for the auto-phasing process.
WI[11]	Second oscillation phase for the auto-phasing process.
WI[12]	Oscillation phase average for the auto-phasing process.
WI[13]	Integer that shows entire estimated sampling time, in microseconds.
WI[14]	Integer (unsigned short) that shows a fraction of estimated sampling time, in microseconds. The value of the estimated sampling time is calculated according to the following formula: WI[13] + WI[14]/2^16.
WI[15]	Address of start of commutation and cogging correction tables.
WI[16]	Address of end of commutation and cogging correction tables.
WI[17]	Length of commutation and cogging correction tables.
WI[21]	Actual length of each recorded vector.
WI[22]	Maximum valid value of SF for current TS value.
WI[23]	Always 0.

Table 3-45: **WI[N] Report Details**

Attributes:	**Type:**	Command/Status report, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Unit modes:	All
	SimplIQ:	Modified


Under most circumstances, this command is used only by the tuning environment.

See also:
WS[N]

WS[N] - Miscellaneous Reports

Purpose:
Provides certain conversion constants and internal states of the drive. WS[N] gives service personnel a fairly comprehensive report, but these details are not normally required for defining an application.

The following summarizes WS[N] reports. Indices omitted from the table are not used.

Index	Report
WS[3]	CPU clock frequency, in Hz.
WS[4]	Width of the PWM frame, in CPU clocks.
WS[5]	A/D bits per one ampere of phase motor current. The A/D resolution is 1/WS[5] ampere.
WS[8]	State of the peak/continuous filter.
WS[10]	Applicable torque limit, in torque command units (see WS[22]).
WS[11]	Returns 1 for drives that support position control, and 0 for drives that support speed control only.
WS[12]	Shoot-through delay, in CPU clocks.
WS[15]	Motor oscillation actual amplitude, for an auto-phasing process.
WS[16]	Reports 1 if previous auto-phasing process failed due to an excessive transfer function phase.
WS[17]	Reports RMS of reference waveform, for automatic current controller tuning.
WS[18]	Reports mean power supply voltage, for automatic current controller tuning.
WS[19]	Reports current following error, for automatic current controller tuning.
WS[20]	Reports the stator field angle, 1024 counts per electrical revolution.
WS[21]	Reports the commutation counter.
WS[22]	Reports the scale between torque commands, in amperes, and their internal representation.
WS[28]	The sampling time of the speed controller.
WS[29]	The basic time quanta for the recorder, in microseconds.
WS[30]	Product capabilities and hardware configuration string. See the following table. The WS[30] report helps the setup program to identify the product. WS[30] has the same format for all *SimpIIQ* products.
WS[33]	Reports the actual peak current saturation in amperes.
WS[34]	Reports the actual continuous current saturation in amperes.
WS[55]	The sampling time of the position controller.
Other indices	Report specifically to the automatic controller tuning process.

Table 3-46: **WS[N] Miscellaneous Reports**

WS[30] is a bit-field.

The bit descriptions of WS[30] are summarized in the following table:

Bits	Meaning	
0…4	**Value**	**Product**
	0	Saxophone
	1	Clarinet
	2	Reserved
	3	Harmonica
	4	Cello
	5	Bassoon
	6	Trumpet
	7	Tuba
	8	Banjo
	9	Cornet
	10	Whistle
	11	Didge
	12	Tweeter
	13	Drum
	14	Reserved
	15	Bee
	16	Hornet
	17	Hawk
	18	Falcon
	19	Eagle
	20	Harp
	21-30	Reserved
5…7	**Value**	**Main Position Sensor**
	0	Reserved
	1	Incremental digital encoder
	2	Resolver
	3	Incremental digital , analog encoder, analog halls, absolute coarse/fine analog encoder
	4	Tachometer or Potentiometer
	5	Absolute position sensor
	6	Reserved
	7	Special treated main feedback look at bits 21..24
8…10	**Value**	**Auxiliary Position Sensor**
	0	Reserved
	1	Incremental digital encoder
	2	Resolver
	3	Incremental digital or analog encoder
	4	Special treated auxiliary feedback look at bits 25..28
	5-7	Reserved
11…15	**Value**	**Grade**
	0	Standard
	1	Advanced
	12-15	Reserved
16	**Value**	**Current Controller**
	0	Analog
	1	Digital

Bits	Meaning
17	**Value CAN Communication** 0 Not present 1 Present
18 - 21	**Value Special Main Feedback Configuration** 0 None 1 Absolute Encoder + Incremental digital/analog interface 2-15 Reserved
22 - 25	**Value Special Aux Feedback Configuration** 0 None 1..:15 Reserved
26 – 31	Reserved

Table 3-47: **WS[30] Drive Personality Descriptor**

Attributes:	**Type:**	Report, Floating
	Source:	RS-232, CANopen
	Restriction:	None
	Range:	Not applicable
	Unit modes:	All

Application note:

WS[20], WS[21]: Brushless DC Commutation

XA[N] - Extra Parameters

Purpose:
Extra filters parameters. This command is not used, code remains for historical SimplIQ compatibility. The extra parameters are now in the PF[N] command.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0
	Default values:	0
	Range:	Previous table
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Canceled

XC, XQ - Execute or Continue Program

Purpose:
Executes the user program from a specified label, or runs a specified function.

- XQ##MYFUNCTION(a,b,c) runs the function MYFUNCTION(a,b,c).
- XQ##LABEL runs from ##LABEL.
- XQ## runs from the start of the user program code.
- The XQ command without a parameter is illegal.
- XQ does not return a value.

The XQ command clears the error status of the program, along with run-time error flags. It does not reset program variables and does not clear the interrupt mask. XC continues a halted program (refer to the HP command).

Attributes:	**Type:**	Command, String
	Source:	RS-232, CANopen
	Restrictions:	Program loaded, compiled and not running. XC requires a running program to be halted.
	Unit modes:	All
	Activation:	Immediate
	SimplIQ	Similar

See also: SimplIQ Programming and Language Guide

XM[N] - X Modulo

Purpose:
Specifies the counting range for the main feedback, which is [XM[1]…XM[2]-1].

The position of the main feedback is always counted cyclically. This means that after the position is counted to its maximum value, the next position count will reset the position counter back to its minimum value. The speed reading is not affected by the position jump. **For example:** If XM[1]=-5 and XM[2]=5, the main position is counted in a cycle length of 10. The main position will always be in the range [-5…4]. If the main feedback rotates in the positive direction, the main position count will proceed from 0, 1, 2, 3, 4 to -5, -4, -3, -2, -1, 0, 1 . . . and so on.

If XM[N] is non-zero with UM=5, the controlled motor position will count cyclically. All motion trajectories will go the short way. **For example:** If XM[1]=-512, XM[2]=512, the initial position is PX=-500 and the target absolute position is PA=500, the PTP trajectory will travel through -500…-512, 511…500 to a total distance of 23 counts.

The cycle length must be positive (XM[2] > XM[1]) and (XM[2] - XM[1]) must be an even number. XM[N] values violating the above conditions will be accepted, but not activated. Until XM[N] can be activated, the motor cannot be started and the parameters cannot be saved in the flash memory (the SV command will not work).

Notes:

- If XM[1] or XM[2] is set so that XM[2] > XM[1] but PX is out of the range [XM[1]…XM[2]], PX will be set to range by taking modulo:
  PX = (PX - XM[1]) mod (XM[2] - XM[1]) + XM[1]
- If a homing process requests a PX value setting out of the range [XM[1]…XM[2]], the request will be ignored and PX will not change.
- If the XM[N] value is selected low and the main speed is too high, more than one full revolution of the main counter may elapse within a single sampling time. This will cause the main position counter to behave unpredictably.
- XM[N] cannot be modified while the homing sequence is active (HM[1] > 0).
- For open loop stepping of rotary motors, XM[2]-XM[1] must be an integer multiple of PN[4], otherwise stepping angle jumps shall appear on position modulo transitions.

Attributes:	**Type:**	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0 MO=1 and SV cannot be used before XM[2] and XM[1] are activated (see "Activation" below)
	Default values:	XM[1]=-5x10^8, XM[2]=5x10^8 (RS), Non-volatile
	Range:	Position range
	Unit modes:	All
	Activation:	XM[2] > XM[1] and XM[2] - XM[1] is an even number
	SimplIQ:	Modified – range changed

See also:
YM[N], PX

YA[N] - Auxiliary Position Sensor Parameters

Purpose:
Defines the behavior and direction of the auxiliary position sensor signals.

The auxiliary encoder pins are multi-role. Their function varies by YA[4]. They may be:

- Inputs for the pulses of the external position incremental sensor or PWM source signals. This mode is used to enable the drive to follow any source of encoder or PWM signals.
- Outputs, repeating the pulses of the main position incremental or analog sensors. This option is only available for digital incremental encoder feedback.
- Output for PWM signal that reflects the current command. The PWM frame will be 2TT[1], and the duty will vary between 0% to 100% when the current command varies from –PL[1] to PL[1]. This mode is useful when several drives need to synchronize their current output.

Index	Description
2	Auxiliary encoder bits per rev (rotary) or per meter (linear)
3	Auxiliary encoder type – 0: Rotary , 1: Linear
4	Auxiliary encoder type: 0: Auxiliary encoder entry used as input for external pulse-and-direction signals 1: Not used 2: Auxiliary encoder entry used as input for external quadrature incremental encoder signals 3: Reserved 4: Reserved 5: 50% duty cycle command format. 0% and 100% duty cycle is interpreted as either maximum positive or negative PWM command depending upon the YA[5] setting. 6: 100% duty cycle command format. Depending upon command polarity signal level, 100% duty cycle is interpreted as either maximum negative or positive PWM command. 7: Auxiliary encoder port used as generator of the PWM signals
5	Auxiliary count direction: 0: Count forward for encoders or P&D inputs. For PWM inputs: forward direction, increasing the duty cycle causes an increase of the controller output command. 1: Count in reverse direction for encoders or P&D inputs. For PWM inputs: reverse direction, increasing duty cycle causes a decrease of the controller output command. YA[5] has no effect if YA[4] specifies that the auxiliary encoder pins repeat outputs (YA[4]=4).

Table 3-48: **XP[N] Entries and Values**

Notes:

- Changing YA[4] resets the position sensor thus resetting the homing position.
- Changing YA[5] does not change PY. It only defines in which direction future encoder pulses will be counted.
- SimplIQ difference: Encoder output is only available for quadrature Main encoder.
- SimplIQ difference: OC[N] runs correctly regardless of the YA[4] state.

Attributes:		
	Type:	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0
	Default values:	YA[1]…YA[3]=0 (RS), Non-volatile YA[4]=2, YA[5]=0 (RS), Non-volatile
	Range:	As in previous table
	Index range:	[1…5]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Modified

See also: OC[N]

YM[N] - Y Modulo

Purpose:
Specifies the counting range for the auxiliary feedback, which is [YM[1]…YM[2]-1].

The position of the auxiliary feedback is always counted cyclically. This means that after the position is counted to its maximum value, the next position count will reset the position counter back to its minimum value. The speed reading is not affected by the position jump. **For example:** If YM[1]=-5 and YM[2]=5, the auxiliary position is counted in a cycle length of 10. The auxiliary position will always be in the range [-5…4]. If the auxiliary feedback rotates in the positive direction, the auxiliary position count will proceed from 0, 1, 2, 3, 4 to -5, -4, -3, -2, -1, 0, 1 . . . and so on.

If YM[N] is non-zero with UM=4, the controlled motor position will count cyclically. All motion trajectories will go the short way. **For example:** If YM[1]=-512, YM[2]=512, the initial position is PY=-500 and the target absolute position is PA=500, the PTP trajectory will travel through -500…-512, 511…500 to a total distance of 23 counts.

The cycle length must be positive (YM[2]>YM[1]) and (YM[2] - YM[1]) must be an even number. YM[N] values violating the above conditions will be accepted, but not activated. Until YM[N] can be activated, the motor cannot be started and the parameters cannot be saved in the flash memory (the SV command will not work).

Notes:

- If YM[1] or YM[2] is set so that YM[2] > YM[1] but PY is out of the range [YM[1]…YM[2]], PY will be set to range by taking modulo:
  PY = (PY - YM[1]) mod (YM[2] - YM[1]) + YM[1]
- If a homing process requests a PY value setting out of the range [YM[1]…YM[2]], the request will be ignored and PY will not change.
- If the YM[N] value is selected low and the main speed is too high, more than one full revolution of the auxiliary counter may elapse within a single sampling time. This will cause the auxiliary position counter to behave unpredictably.
- YM[N] cannot be modified while the homing sequence is active (HM[1] > 0).

Attributes:		
	Type:	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	MO=0 MO=1 and SV cannot be used before YM[2] and YM[1] are activated (see "Activation" below)
	Default values:	YM[1]=-5x10^8, YM[2]=5x10^8 (RS), Non-volatile
	Range:	Position Range
	Unit modes:	All
	Activation:	YM[2] > YM[1] and YM[2] - YM[1] is an even number
	SimplIQ	Modified, range changed

See also:
XM[N], PY

ZX[N] - User Program and Auto-tuning Temporary Storage

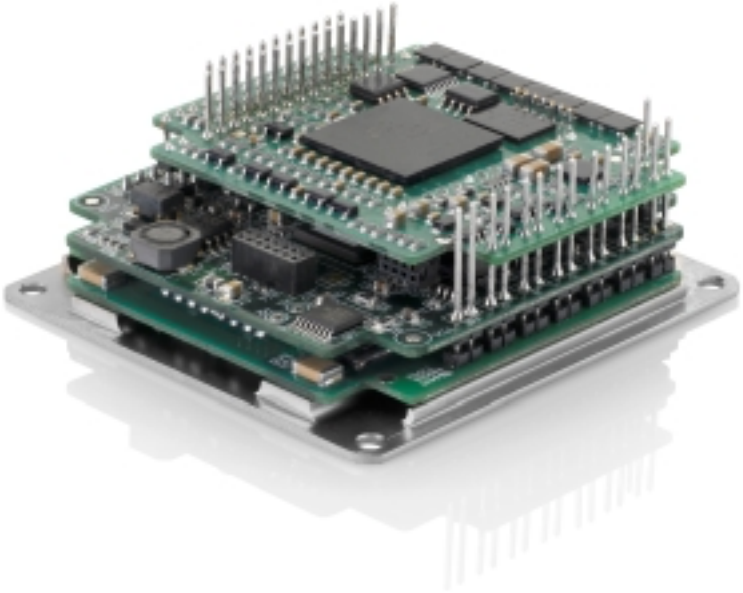
Purpose:
Serves as temporary storage of reference waveforms for the controller in automatic tuning experiments. Out of the scope of auto-tuning experiments, the ZX[N] vector can be used as a large temporary storage for user programs.

Notes:

- The elements of ZX[N] are *short* (16-bit) integers.
- The ZX[N] command truncates long numbers to shorts in the range [-32767 … 32767].

Attributes:		
	Type:	Parameter, Integer
	Source:	Program, RS-232, CANopen
	Restrictions:	None
	Default value:	0
	Range:	[-32,767…32,767]
	Index range:	[0…1023]
	Unit modes:	All
	Activation:	Immediate
	SimplIQ:	Similar

SimpIIQ for Steppers
Application Note



Ver. 1.1 - June 2009

Elmo
Motion Control

DFT

Important Notice

This guide is delivered subject to the following conditions and restrictions:

- This guide contains proprietary information belonging to Elmo Motion Control Ltd. Such information is supplied solely for the purpose of assisting users of *SimpIIQ for Steppers* servo drives.
- The text and graphics included in this manual are for the purpose of illustration and reference only. The specifications on which they are based are subject to change without notice.
- Information in this document is subject to change without notice.

Doc. No. MAN-STECR

Revision History

Ver. 1.1	June 2009	Updates
Ver. 1.0	Feb 2008	First revision.

Elmo Motion Control Ltd.	**Elmo Motion Control Inc.**	**Elmo Motion Control GmbH**	
64 Gisin St., P.O. Box 463 Petach Tikva 49103 Israel	42 Technology Way Nashua, NH 03060 USA	Steinkirchring 1 D-78056, Villingen-Schwenningen Germany	
Tel: +972 (3) 929-2300 Fax: +972 (3) 929-2322	Tel: +1 (603) 821-9979 Fax: +1 (603) 821-9943	Tel: +49 (0) 7720-85 77 60 Fax: +49 (0) 7720-85 77 70	
info-il@elmomc.com	info-us@elmomc.com	info-de@elmomc.com	**www.elmomc.com**

Contents

Chapter 1:Introduction

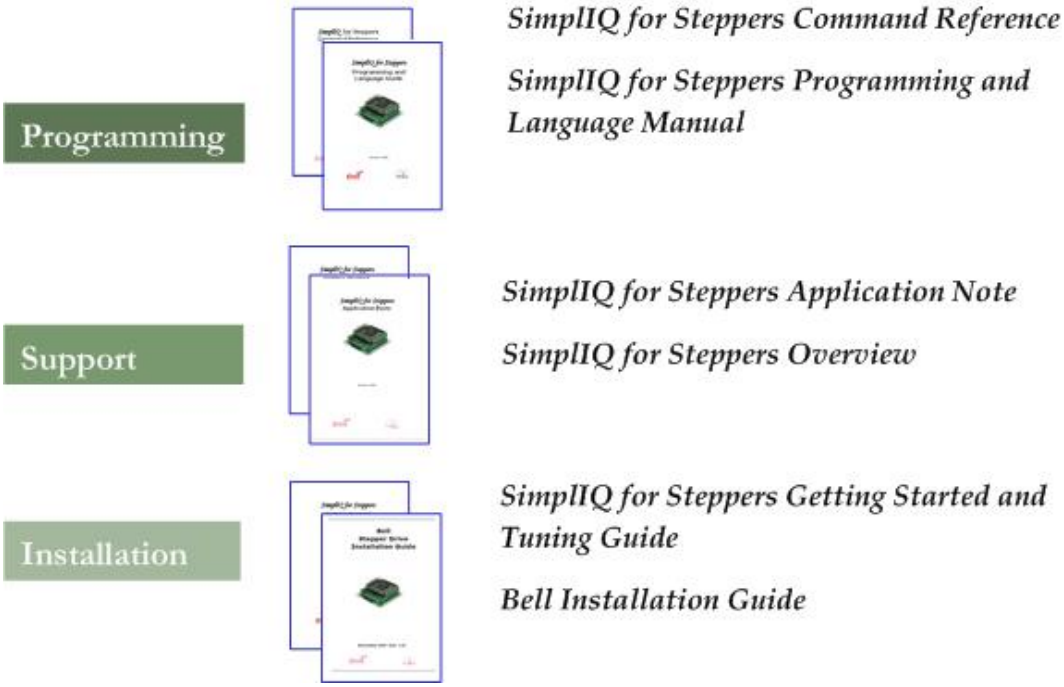
These Application Notes explain the motion control and operating issues of SimplIQ for Steppers, particularly issues that are beyond the scope of the Command Reference.

Instead of explaining each command separately, this document describes a topic, and the interaction between the various commands and parameters that may be involved.

SimplIQ for Steppers offers state-of-the-art control algorithms, integrated with tuning tools.

The complexity of control algorithms requires the use of the software tuning tools that are shipped with SimplIQ for Steppers. Understanding the underlying control structure will enable you to use these tools much more effectively.

The diagram below shows the SimplIQ for Steppers documentation set:



Chapter 2:Device and Software Organization

This chapter describes the internal structure of the SimplIQ for Steppers. Please note that SimplIQ for Steppers may have several hardware implementations, and the issues discussed in this section may be dependent on the specific implementation.

2.1. The SimplIQ for Steppers Processor and the Motion Processor

Some SimplIQ for Steppers implementations include a separate motion processor, as shown below.

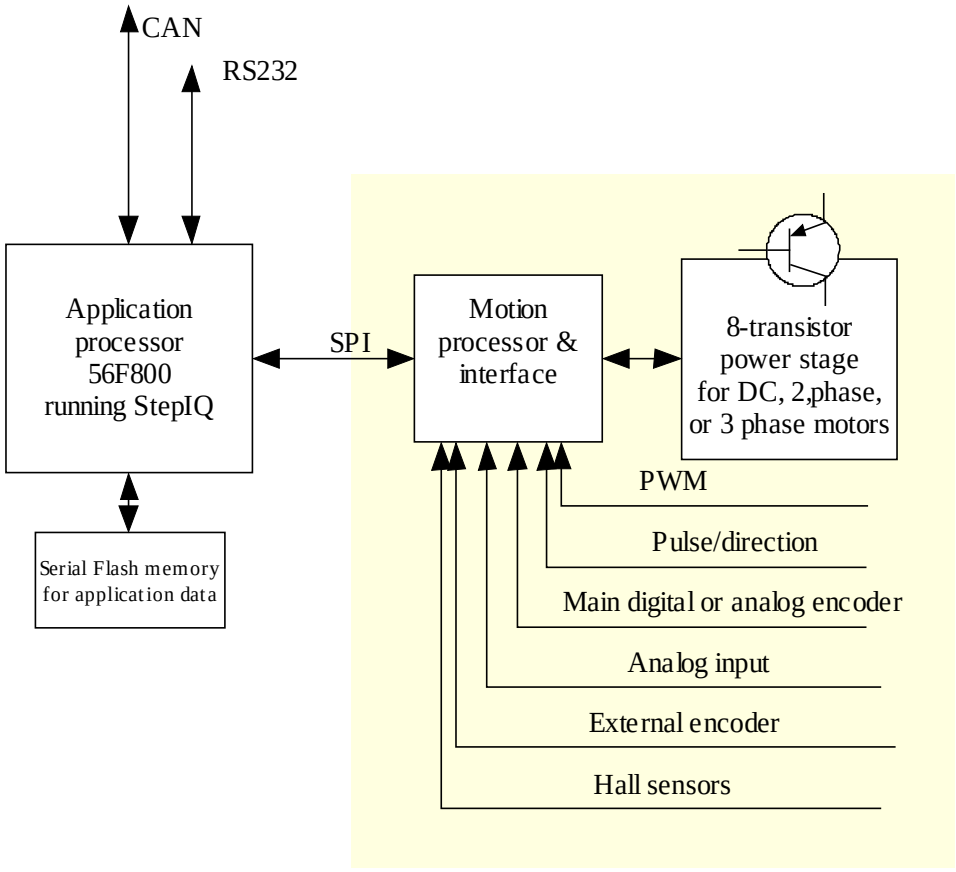


Figure 2-1: **SimplIQ for Steppers processor and a motion processor**

The SimplIQ for Steppers processor runs the command interpreter for all the communication and programming types, and generates the command reference for the motion processor.

The motion processor interfaces with the sensor and the power electronics, and calculates the real-time control.

The exception is the synthesis of speed and current commands from analog and pulse input data. This synthesis is carried out by the motion processor, to allow minimum delay.

The cycle time of the SimplIQ for Steppers processor is fixed and will normally be 360 µsec. The WI[4] command returns this value.

The TT[1] command sets the sampling time for the motion processor.

2.2. Storing Non-volatile Application Data

SimplIQ for Steppers stores application data permanently in flash memories.

This data includes:

- Controller, sensor, and IO parameters
- The user-defined program
- Cogging and commutation correction tables.

All this data is stored in a serial flash which ensures that the application data survives minor firmware updates (updates that do not change the database structure).

The Controller, sensor, and IO parameters, as well as the correction tables, are saved by the SV command, and restored by the LD command.

The user program uses a special flash segment. This assures that user program editing can take place without risking the controller, sensor, and IO parameters. User program maintenance includes the CP, CC, DL and LP commands. User program downloading detail is described in the User Program Manual.

The cogging and the commutation correction tables present a special challenge. These are large data blocks that need be accessed in real-time by the motion processor. For this reason, the correction tables have another copy stored in the flash memory of the motion controller.

The two copies of the correction tables can be synchronized by the SI command.

Correction tables synchronization is checked on power-on: if the two copies of the tables do not match, motion is disabled. The correction table synchronization is only checked once. If the user changes the correction tables in the SimplIQ for Steppers database, but does not synchronize the motion processor, this change will have no effect on the motion.

Chapter 3:Unit Modes

The *SimpIIQ* drive's feedback can be structured in a number of different ways. These options are called "unit modes" and are programmed by the UM parameter. The unit mode can be switched only with the motor turned off, because the feedback structure must be rearranged for each mode separately. The following unit modes are available:

Value	Description (Related Commands)
0	Open loop voltage
1	Torque control mode
2	Speed control mode
3	Micro-stepper open loop mode
4	Dual feedback position control mode
5	Single feedback position control mode

Table 3-1: **Unit Modes**

3.1. Unit mode 0: Open loop voltage control

Voltage control mode
In this mode, the motor voltage is set directly by the VC software command. This mode is useful only for very special cases, such as motors with very high resistance and very short mechanical time constants.

Using this mode without extreme care may damage the motor.

3.2. Unit Mode 1: Torque Control

In this mode, the drive controls the motor torque only. The drive may serve as a torque driver, controlled by an external controller.

If position sensors are connected, *SimpIIQ* evaluates the speed for over-speed protection, as specified by parameters LL[2] and HL[2].

The torque command may be set by a software command, an analog input or a combination of the two, according to the following figure:

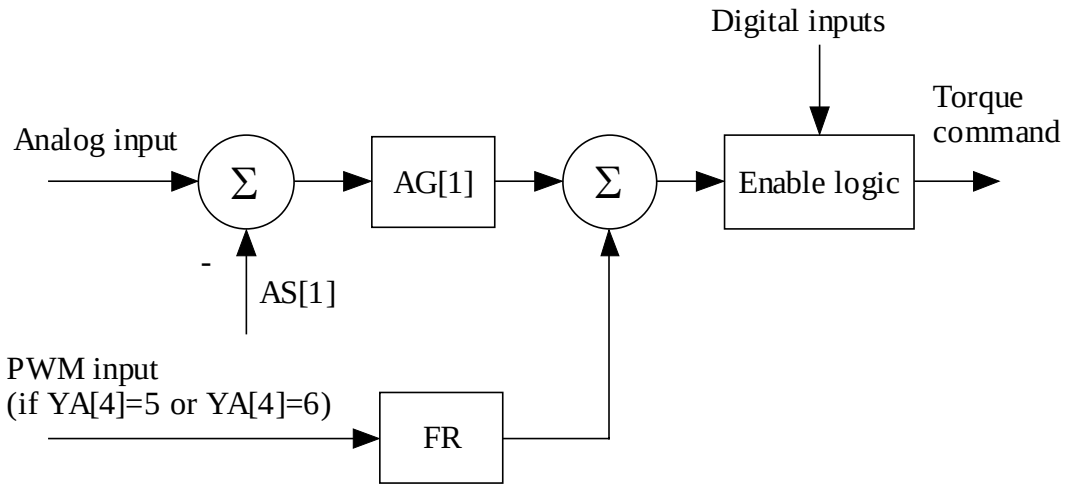


Figure 3-1: **Unit Mode 1 (Torque) Structure**

The AS[1] parameter compensates for possible offsets in the driving equipment, and internally in the *SimpIIQ* drive. If you do not use the analog input for the torque command, set AG[1] = 0 or RM = 0 in order to avoid the noise and offset that affect the drive torque command.

The combined (software and analog input) current demand is reported by DV[1].

The enable logic does the following:

- If hard stop is active, sets the torque command to zero
- If RLS is active, clips negative torque commands to zero
- If FLS is active, clips positive torque commands to zero.

The Stop Manager does not affect the reference generator. When the relevant switch is released, the torque command value set by the reference generator is restored.

The external reference to the current loop is implemented directly in the motion processor, thus is free from any inter-processor communication delay.

3.3. Unit Mode 2: Speed Control

In this mode, the drive controls the motor by speed feedback.

The reference generator issues a speed command and the speed controller uses the speed command to demand torque from the current controller.

The reference to the speed controller is a sum of software commands and an auxiliary speed command that is derived using the analog input, the auxiliary encoder input or PWM.

The Stop digital input and the limit switches (RLS, FLS) can be used to stop or to limit the direction of the motor (described in section 3.3.2). The drive will abort upon over-speed, as specified by the parameters LL[2] and HL[2]. The speed control scheme is shown in the figure below:

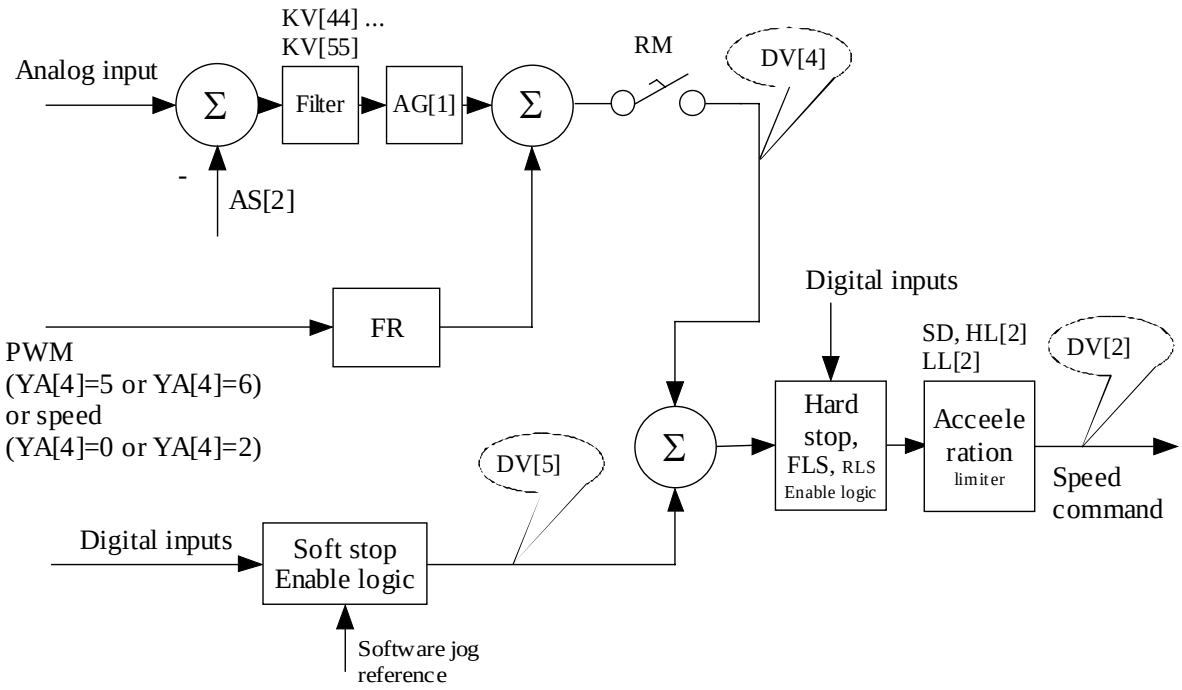


Figure 3-2: **Unit Mode 2 (Speed) Structure**

The DV[2] command reports the combined (software and analog input) speed demand, after being further processed for acceleration and speed limiting, and for the switch actions RLS, FLS and STOP.

DV[4] and DV[5] retrieve the external and software components of DV[2], respectively.

The analog or PWM input is most useful when the *SimpIIQ* drive serves as an inner controller, embedded in an external control loop. The auxiliary encoder speed input enables the drive to issue speed commands relative to a conveyor or other moving object. If you do not use the analog input for the torque command, set RM = 0, in order to avoid noise.

The filter is optional. You can design a linear arbitrary filter of an order up to 4, and use it to filter the analog input for extra smoothness.

The auxiliary speed reference is generated according to the block diagram that follows. The parameters relevant to auxiliary speed command generation are:

Parameters	**Description**
AG[2]	Analog input gain, counts/second/volt
AS[1]	Analog offset
FR[2]	Follower gain
KV[44]…KV[55]	Filter for analog input – refer to the KV command in the command reference

Parameters	Description
RM	Reference mode: 1: Use auxiliary speed command 0: Null auxiliary speed command

Table 3-2: **Auxiliary Speed Command Parameters**

3.3.1 Software Speed Command

The software speed command generator generates speed commands, subject to acceleration, deceleration and speed limits. The following commands are relevant to the generation of the software speed command:

Command	Description
AC	Profiler acceleration limit, in counts/second^2.
BG	Begin command.
DC	Profiler deceleration limit, in counts/second^2.
JV	Profiler final speed command, in counts/second.
PM	PM=1: Normal acceleration limits form software speed command. PM=0: AC and DC are ignored and SD is used instead. The Composer setup program uses this mode when tuning the speed controller. During regular operation, PM=1 is recommended.
SF	Smooth factor: time, in milliseconds, required to develop the full acceleration of AC and deceleration of DC.
VH[2]	Maximum speed command, in counts/second.
VL[2]	Minimum speed command (bound by speed command, negative value), in counts/second.
HL[2]	Maximum allowed speed, in counts/second. If feedback indicates a higher value, the motion is automatically aborted.
LL[2]	Minimum allowed speed, in counts/second. If feedback indicates a lower value (high value in negative direction), motion is automatically aborted.
IL[N]	Map functions to digital inputs, which may function as hardware ST or BG instructions.

Table 3-3: **Software Speed Commands**

The BG command enters a new desired speed value together with the acceleration and deceleration limits. The AC acceleration is used to increase the absolute speed value, while the DC deceleration is used to decrease the absolute speed value.

The algorithm of the acceleration-limited software speed command is:

1. Has any new BG command been accepted by the software or hardware? If yes, update the speed target to the value of JV, and also update the permitted acceleration and deceleration to the values of AC and DC.

2. If the speed target and speed command are positive, and the speed target is greater than the speed command, select AC for the acceleration limit. If the speed target and speed command are negative, and the speed target is less than the speed command, also select AC. Otherwise, select DC for the acceleration limit.

3. Converge the speed command towards the speed target as fast as acceleration/deceleration limits permit. If the speed command already equals the speed target, do nothing.

The SF (Smooth Factor) determines the time required to build the full acceleration.

Example 1:
Let:
`MO=1; JV=4000; AC=100,000; DC=200,000; SD=1e6; PM=1; RM=0; SF=0; BG;`
The following figure depicts how the speed command to the controller (`DV[2]`) tracks the target speed specified by changing the `JV` parameter, followed by a `BG`.

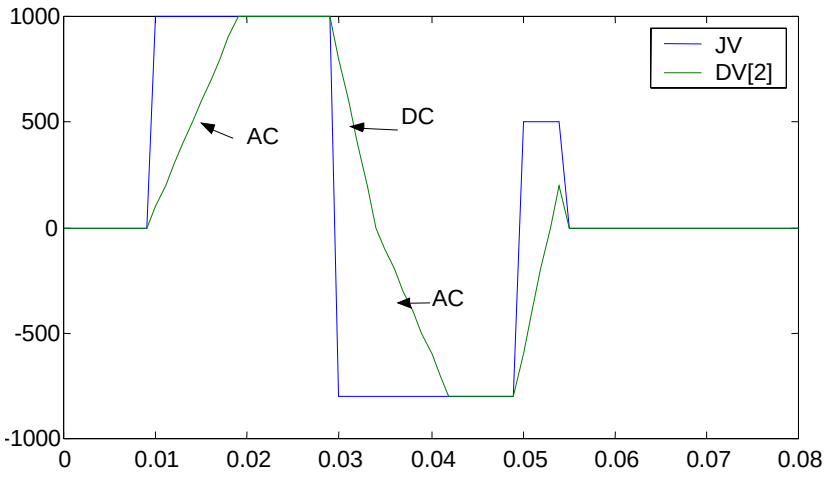


Figure 3-3: **Speed Profiling Using JV, AC and DC**

The speed reference to the controller may be changed at any time, regardless of the state of the profiler. In the previous figure, the required speed was changed at time 0.065, from -1000 to 0, before reaching the speed command of -1000.

Example 2:
This example demonstrates the smoothing filter and the smoothing factor SF:
`MO=1; JV=4000; AC=100,000; DC=100,000; SD=1000000; PM=1; RM=0;BG`
SF has three different values:

- The SF=0 graph displays sharp corners (discontinuous acceleration), because there is no smoothing.
- The SF=10 graph takes 10 milliseconds more to stabilize the speed software command; however, the speed reference profile is much smoother.

- For the SF=50 graph, the smoothing is so strong with respect to total acceleration time that the AC acceleration is never reached.

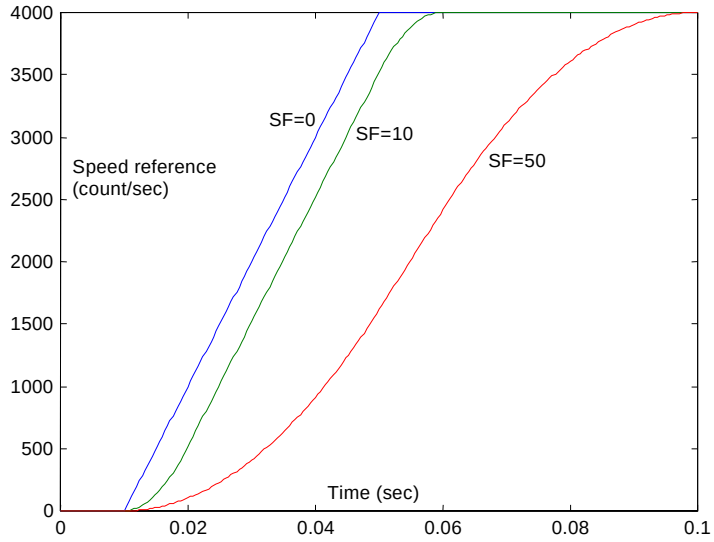


Figure 3-4: **Speed Command for Different Smooth Factors**

The profiler smoothes the motion and decreases (and may also prevent) overshoots. The price is the introduction of delay in the controller response.

3.3.2 Acceleration Limiting and Switch Logic

The limiting logic performs the following functions:

- Brings the motor to a stop upon a software or hardware ST command.
- Brings the motor to a stop when the speed demand is positive and FLS (Forward Limit Switch) is active.
- Brings the motor to a stop when the speed demand is negative and RLS (Reverse Limit Switch) is active.
- Prevents acceleration or deceleration beyond the SD (stop deceleration) limits.

The parameters relevant to the stop manager are:

Parameters	Description
SD	Maximum motor acceleration/deceleration, in counts/second^2
VL[2], VH[2]	Speed command limit
IL[N]	Input logic: defines digital inputs as hard stop or as direction limit switches (RLS or FLS)

Table 3-4: **Stop Manager Parameters**

The acceleration limiter prevents the speed controller command from changing abruptly by limiting the rate of reference change to SD counts/second^2. When the motor stops due to a switch action, a zero command is placed at the input to the acceleration limiter. The motor is brought to complete stop using the SD deceleration.

When the switch action terminates (a Stop switch is released, for example), the SD acceleration applies until the actual speed command converges to the speed demand.

The following rules are always kept:

- The total speed command is limited to the range [VL[2]…VH[2]].
- When an event (RLS, FLS, etc.) forces the zero speed command, this event does not affect the reference generator, which continues to behave normally. When the switch action is complete, the speed command converges to the output of the reference generator.
- The acceleration limiter limits the decelerations and accelerations due to abrupt changes of the auxiliary reference. The SD applies to the sum of the software and the auxiliary speed references. The AC and DC parameters are only relevant to software commands.
- When AC and DC are greater than SD, they become irrelevant, because SD further limits acceleration.
- The IL[N] command can program a variety of stop options to an input pin. A pulse at a programmed I/O pin can be setup to stop the software reference generator, or to zero the entire speed command.

Example:

```
SD=100,000, MO=1, AC=10,000, DC=10,000, JV=5000, BG;
```

At the time of 0.2 seconds, a switch programmed as a hard stop is activated, and released at the time of 0.4 seconds. The following waveforms result:

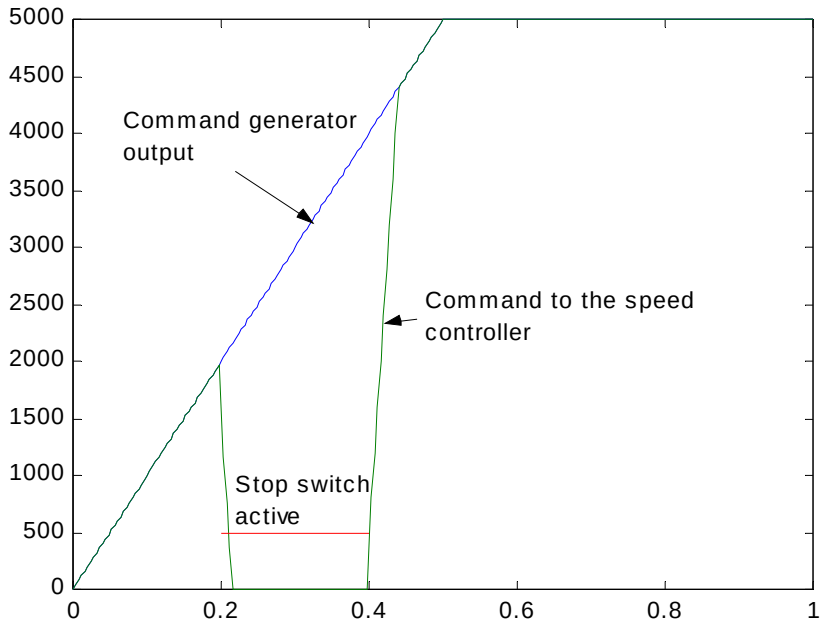


Figure 3-5: **Results of a Sample Stop Switch Activation**

When the stop switch is applied, the speed command decreases to a complete stop, decelerating at 100,000 counts/second^2. When the switch is released, the acceleration of 100,000 counts/second^2 is used to converge to the reference generator command.

3.4. Unit Mode 3: Open & Closed Loop Stepper Control

This chapter describes how a *SimpIIQ* device works with one of the following options:

- Classical open-loop sensorless stepper drive.
- Stepper motor with position feedback.

The PN[9] parameter determines which option is used.

The motor may be two-phase or three-phase: the contents of this chapter apply equally.

For both of the modes, the motor current level is a fixed level with additions. For the speed and the acceleration of position reference waveform – see the HT[N] command.

The motor steering is made by micro-stepping the electrical angle of the stator field. The rotor is supposed to track the stator field angle. This is possible if the electrical field angle varies slowly enough.

The benefit of this unit mode is that it does not require a commutation sensor. In the open loop mode, no position sensor is required at all.

3.4.1 Open Loop Drives

The benefits of an open loop drive are:

- It does not require any position sensor.
- It does not require complex motor tuning.
- It provides smooth motion at low speed.

The main drawbacks are:

- Low efficiency, since the holding current exists regardless of the mechanical output torque.
- Tendency to oscillate. The stepper torque is proportional to the position error, so that it generates a harmonic oscillator with the natural frequency of $\omega_r = \sqrt{\frac{K_T I}{J}}$.

  Here K_T is the motor torque sensitivity, I is the current, J is the motor and load inertia, reflected to the motor.
- When the field is driven too fast, the stepper stalls. This means that it does not generate DC torque or power anymore – it simply vibrates in place.

3.4.2 Closed Loop Drives

In the closed loop mode, a position sensor is used to close position feedback.

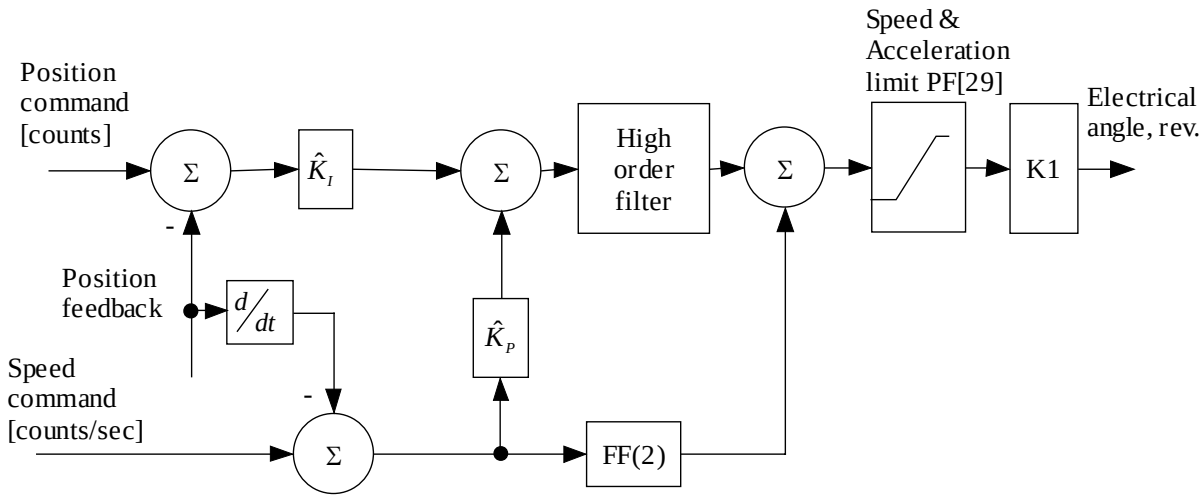
The benefits of closed loop stepper drive are:

- It does not require any commutation sensor – there is no need to tune the commutation, and there may even be flexibility or a play between the position sensor and the motor shaft.
- The position static error is reduced to zero.
- It provides smooth motion at low speed.

Its main drawbacks are:

- Low efficiency, since the holding current exists regardless of the mechanical output torque.
- Tendency to oscillate if the closed loop controller is tuned to low bandwidth. The stepper torque is proportional to the position error, so that it generates a harmonic oscillator with the natural frequency of $\omega_r = \sqrt{\frac{K_T I}{J}}$. Here K_T is the motor torque sensitivity, I is the current, J is the motor and load inertia, reflected to the motor. When tuned to high bandwidth (this requires a good quality position sensor and no play), the stepper's natural resonance is almost completely eliminated.
- When the acceleration limits are not set correctly (PF[29]), the stepper stalls. This means that it no longer generates DC torque or power – it simply vibrates in place.

The position controller structure is depicted below.



- The parameter KF[29] tunes the acceleration limiter, which prevents the controller from demanding too great accelerations, as too great accelerations may cause a stepper to stall. This parameter sets the ratio between the current level in [A] and the permitted drive acceleration in encoder counts/sec^2.

- The controller parameters are $\hat{K}_P = 0.001KP[2]$, $\hat{K}_I = 0.001KI[2]$ if gain scheduling is not used (GS[2]=0) or equivalently defined on the KG[N] parameters if gain scheduling is used.
- Normally one should set FF[2]=1.
- For closed loop stepper mode, use PN[4]=1024.
- The parameter K1 = CA[19]/CA[18] converts encoder counts to electrical revolutions.
- The high order filter is defined in the chapter on filters.

3.4.3 The Stepper Mode Position and Current Command Generator

The position and the torque command in the stepper mode is shown in the figure below.

The position command, generated in a similar way to other position modes, depends on the value of PN[9]:

- If PN[9]=0 (open loop) the position command is actually a reference for the electrical angle.
- If PN[9]=1 (closed loop) The position command is a reference for the closed position control loop.

For position command generation, refer to the Position Reference Generator.

The HT[N] command sets the torque ; torque additions may be by analog input (refer to the AG[1], AS[1], and RM commands).

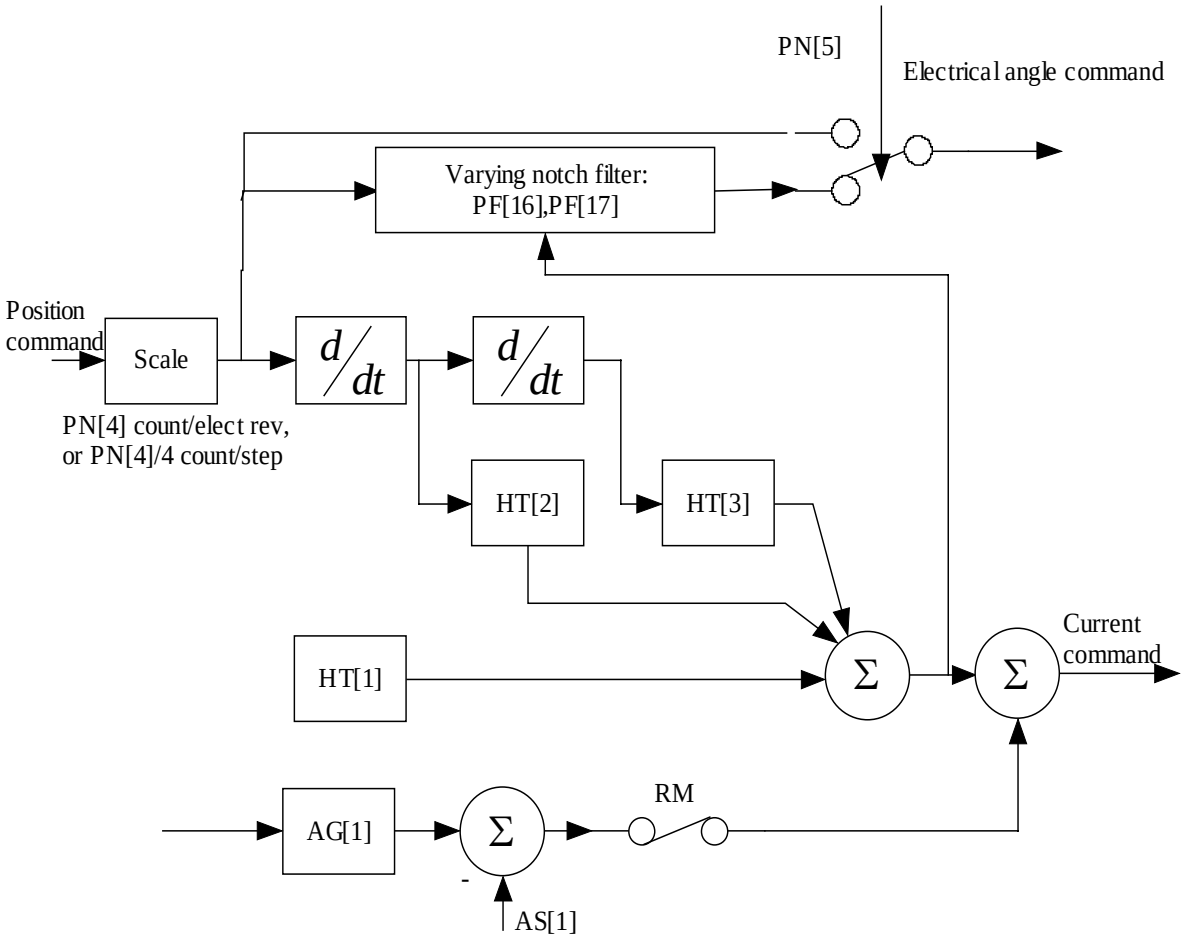


Figure 3-6: **Torque and position reference in the open loop stepper mode**

Note the following:

PN[4] sets the scale between the position command and the electrical angle rotation. PN[4] may be viewed as the "micro-stepping ratio": There are PN[4]/4 position counts per step.

For open loop stepping of rotary motors, XM[2]-XM[1] must be an integer multiple of PN[4], otherwise stepping angle jumps will appear on position modulo transitions.

HT[1] is the holding torque in Amp: the amount of current in the motor when its position is steady.

HT[2] and HT[3] enable the insertion of extra torque to compensate for increased speed and acceleration. The units of HT[2] are in Amp/(Count/Sec) and the units for HT[3] are in Amp/(Count/Sec^2).

HT[2] and HT[3] are defined differently for open loop and closed loop stepper control (see PN[9]).

- For open loop mode, HT[2] and HT[3] are defined for PN[4] counts/electrical rev. or PN[4] × S/4 counts/mechanical rev. where S is the number of steps.
- For closed loop mode, HT[2] and HT[3] are defined for the encoder counts.

The varying notch filter is for compensating the oscillations that follow from open loop operation. It notches the frequency $\omega_r = \sqrt{\frac{K_T I}{J}}$ out, using the inertia in PF[16], the K_T of PF[2], and the damping of PF[17].

The current *I* in the calculation of ω_r does not account for current driven by analog input.

3.5. Unit Mode 4: Dual Feedback Mode

Dual feedback mode is used when different sensors are used for speed/commutation and for position. This mode is commonly used when the motor drives the load through a reduction gear. The controlled position is that of the load. The inner loop controls speed, with much higher bandwidth.

A sensor mounted directly on the motor allows much better control bandwidth than a load sensor. This is because the motor sensor is much less susceptible to delay and backlashes.

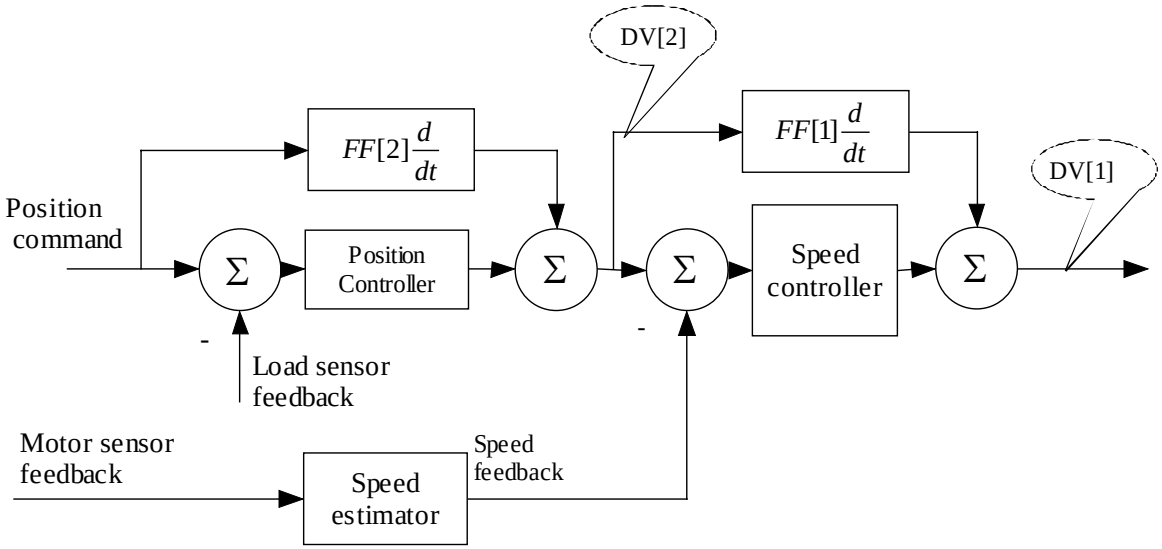


Figure 3-7: **Dual Feedback Mode (UM=4)**

For further details:

The Position Reference Generator

The Position controller

The Speed controller

The speed command, multiplied by the gain FF[2], is fed as reference to the speed controller in addition to position correction. Setting FF[2] exactly to the gear ratio between the position sensor and the speed sensor minimizes the tracking errors.

3.6. Unit Mode 5: Single Feedback Mode

Single feedback mode is used when the same sensor is used for speed, commutation and position. This setup is quite common and it is used whenever a separate load sensor is not required.

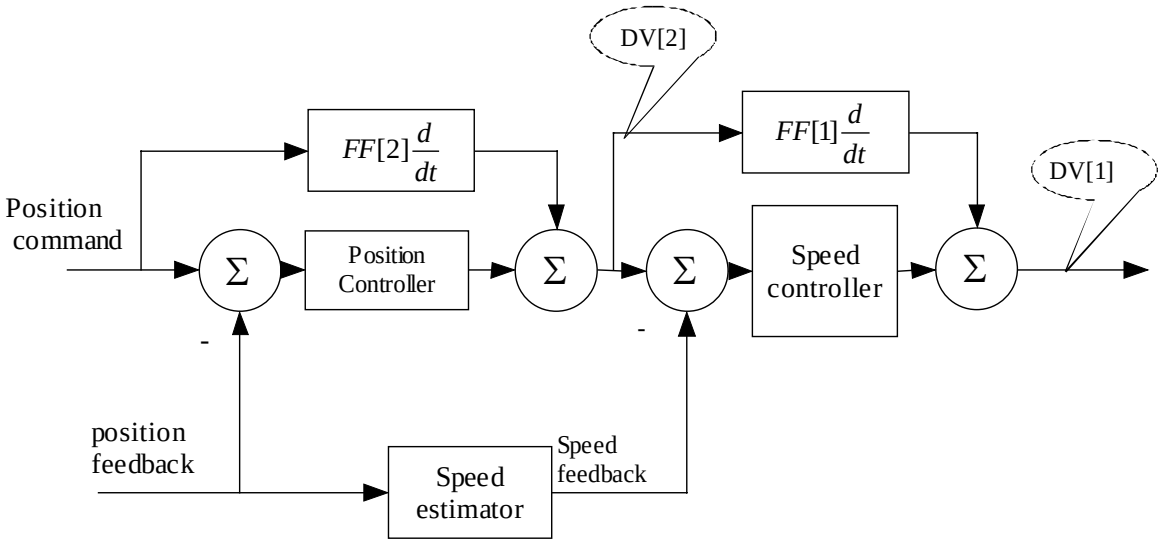


Figure 3-8: **Single Feedback Mode (UM=5)**

The Position Reference Generator

The Position controller

The Speed controller

The speed command, multiplied by the gain FF[2], is fed as reference to the speed controller in addition to position correction. Setting FF[2] = 1 minimizes the tracking errors.

Chapter 4:Commutation and Pole Identification

4.1. General Description

This chapter does *not* apply to DC brush motors

Commutation parameters are normally set using the support software.

The *SimpIIQ* drives fixed magnet motors. The fixed magnet interacts with the magnetic field created by the motor windings.

If the winding's field is kept in a constant direction, the motor will come to a steady state with the windings field aligned with the fixed magnets.

Mathematically:

$T = K_T I \cos(\theta_E - \theta_F)$

where:

θ_E is the rotor electrical angle. For brushless motors $\theta_E = \frac{p}{2}\theta_M$, where θ_M, p are the shaft angle and the number of pole pairs respectively. For steppers, $\theta_E = \frac{n}{4}\theta_M$ where n is the number of steps.

θ_F is the "transformation angle" - the electrical angle of the windings field.
$\theta_E - \theta_F$ is the angle between the magnet and the field of the winding.
K_T is the torque sensitivity. It is proportional to the strength of the magnet.
T is the motor winding torque.
I is the motor current.

The *SimpIIQ* drives can set the transformation angle θ_F freely. The process of selecting the transformation angle is called **commutation**.

Use the parameter CA[25] to set the direction of field movement when θ_F increases.

Parameter	Description
CA[25]	Motor direction: 0: Keep the original motor direction, as connected by the user. 1: Reverse phase driving. The effect of CA[25] is to set the direction of torque/force for positive current demand.

 With closed loop control, CA[25] and CA[16] (sensor direction) must match so that positive current will drive the feedback sensor in the positive direction.

4.1.1 Stepper Commutation

With stepper commutation, the transformation angle is set to point at the desired rotor position. The drive need not be informed where the rotor is; it simply assumes that the rotor will come to rest at the field position.

Stepper commutation is simple and reliable. Its main drawback is that normally $|\theta_E - \theta_F| << 90^o$; therefore, large currents are required to generate a given torque. In its steady state, the motor torque is zero and not affected by motor current. The sensitivity of motor torque to deviation of the rotor angle is maximal. The great sensitivity of the torque to rotor angle generates a fast, but oscillatory position feedback.

4.1.2 Vector Commutation

With vector commutation (for 3-phase or 2-phase motors), the windings field is set to point 90^o away from the rotor position and the drive must know where the rotor is in order to maintain this field direction.

The advantage of vector commutation is its maximum torque per given motor current, and its smooth, controllable torque. It is ideal for servo applications. Vector calculations involve a considerable amount of real-time calculation and requires a rotor position sensor. The torque is not sensitive to rotor angle.

Most brushless motors have two or three phases (coils, or windings).

The *SimpIIQ* drive supports both types.

For three-phase motors, the phases are called A, B and C.

For two-phase motors, the phases are called A and B .

When the rotor travels, the transformation angle rotates. The coils of the three-phase motor are powered in the sequence A-B-C-A-B-C . . . (or A-B-A-B... for two phase motors) and so on.

You can read the electrical and mechanical angle of the motor using the following commands:

Command	Description
WS[20]	Stator field angle, in 1024 counts/revolution units. Stator field angle (degrees) = WS[20] x (360/1024).
WS[21]	Commutation counter. WS[21] counts the main high-resolution position sensor, modulo CA[18].

 For analog encoders, WS[21] counts the position with maximum interpolation, regardless of the user selected interpolation factor CA[31]

The process of initializing the commutation counter WS[21] is called commutation finding, or pole identification.

Commutation finding is made when the motor is first set to ON. The process of commutation finding may involve extra motions, according to the method you select – refer to the next sections of this chapter. The PN[6] parameter determines if commutation search is made every motor ON or just the first time.

4.2. Commutation Sensors

Vector commutation requires rotor position sensors. Commutation sensors are divided into two main groups:

- Direct field sensors that sense the magnetic field of the motor
- Shaft position sensors

 Commutation with position sensor is always made with the main sensor, even if the auxiliary sensor is the position feedback (UM=4).

4.2.1 Hall Sensors

The Hall sensors report the rotor electrical position directly. However, digital Hall sensors provide only rough information with six counts per electrical cycle.

SimpIIQ can use Hall sensors as the only information for commutation. Many applications achieve smoother commutation by using digital Hall sensors for initial pole identification and for backup; most of the time the commutation is made by a higher-resolution position sensor.

The following table describes the most common digital Hall sensor arrangement.

Hall A	Hall B	Hall C	Electrical Rotor Position (Degrees)	Transformation angle for best torque (Degrees)
0	0	0	Illegal	
1	0	0	330 – 30	90
1	1	0	30 – 90	150

Hall A	Hall B	Hall C	Electrical Rotor Position (Degrees)	Transformation angle for best torque (Degrees)
0	1	0	90 – 150	210
0	1	1	150 – 210	270
0	0	1	210 – 270	330
1	0	1	270 – 330	30
1	1	1	Illegal	

Table 4-1: **Hall Sensor Arrangements (1)**

There are many other Hall sensors arrangements; they differ in sensors polarity, and offset.

SimpIIQ drives are very flexible in accepting Hall sensors; they may be connected with every order, polarity and offset.

The following parameters define Hall sensors arrangement:

Command	Description
CA[1]	Digital Hall sensor A polarity (1 for active high, 0 for active low).
CA[2]	Digital Hall sensor B polarity (1 for active high, 0 for active low).
CA[3]	Digital Hall sensor C polarity (1 for active high, 0 for active low).
CA[4]	Actual Hall sensor connector to Hall A connector pin: 1 for A, 2 for B and 3 for C.
CA[5]	Actual Hall sensor connector to Hall B connector pin: 1 for A, 2 for B and 3 for C.
CA[6]	Actual Hall sensor connector to Hall C connector pin: 1 for A, 2 for B and 3 for C.
CA[7]	Offset of digital Hall sensors. The Hall sensor transition electrical angles are [30,90,150,210,270,330] deg + CA[7]x360/1024 The units are 1024/electrical revolution.
CA[20]	Digital Hall sensors present: 0: No digital Hall sensors connected 1: for Digital Hall sensors connected

Table 4-2: **CA Vector - Digital Hall Sensor Arrangement Parameters**

The parameter CA[17] defines how Hall sensors are used for commutation:

Command	Description
CA[17]	0: Use only Hall sensors. The electrical angle resolution will be 60 electrical degrees. 1: Hall sensors are used together with a high resolution sensor. On the first Hall sensor transition, the exact electrical angle is locked and commutation continues by position sensor increments. The Hall sensors continue to monitor commutation correctness. The parameter PA[1] defines the tolerance for this monitoring. 2: Halls are the sole commutation source. On high speeds, use time-based interpolation for smoothing the electrical angle estimate. 8: Every Hall sensor transition, the exact electrical angle is locked and commutation continues by position sensor increments until the next Hall sensors transition. This mode is useful when there is a play, slippage, or flexibility between the high resolution sensor and the motor.

4.2.2 Aiding Commutation with High Resolution Position Sensors

SimpIIQ can use any main position sensor to aid commutation. These sensors normally have good resolution, but must be homed (referenced absolutely) with respect to the rotor electrical angle.

For commutation calculation, we must know how many bits the shaft sensor counts per single electrical cycle. If this number is not an integer, the calculated commutation angle may accumulate numerical errors and cause the motor to lose torque. This is not a serious issue with linear motors that travel limited distance.

The parameters for high resolution commutation are:

Command	Description
CA[16]	Sensor direction. 0: Use normal 2: Invert direction Note that the sensor direction is also used for control feedback, so set CA[16] according to your application needs and match CA[25] so that driving positive current will rotate the motor in the positive direction.

Command	Description
CA[18]	Feedback bits ("counts") per revolution, after resolution is multiplied by 4, in the range [6…530,000,000]. - For **Analog Encoders**, the resolution for commutation is always taken as x2^{16}, regardless of CA[31]. - For Analog Hall sensors always use CA[18]=65536 - For systems with **Halls only**, CA[18] is calculated as CA[19] * 6
CA[19]	The number of feedback counts per electrical revolution is CA[18]/CA[19]. For brushless motors, this is the number of pole pairs. For steppers, this is one quarter of the steps count.
CA[21]	Position sensor present: 0: No high-resolution commutation sensor. Commutation will be based on digital Hall sensors only. 1: Main position sensor will be used for commutation.

For rotary motors, each mechanical shaft rotation involves an integer number of encoder counts, per an integer number of electrical cycles. This means that the commutation can be kept accurate using:
WS[21] = mod (encoder, CA[18])

And the transformation angle is $WS[20] = \frac{rem\left(WS[21], CA[18]/CA[19]\right)}{CA[18]/CA[19]} \times 360^{o}$

Examples:

- For rotary Quadrature **Incremental Encoders** with 1000 lines, CA[18] is 4000. If the motor is linear, CA[18] reflects the electrical cycle.
- For linear encoder with 1000 lines/m (4000 counts/m) with the distance between pole sets is 0.1 m, set CA[18] is 400, CA[19]=1. In this example, CA[18] could be set as any multiple of 400, such as CA[18]=800,CA[19]=2, or CA[18]=1200,CA[19]=3.
- For Analog encoders, *SimpIIQ* uses the full resolution for commutation. This means 2^{16} counts per cycle (line), regardless of the interpolation setting of CA[31]. For example, a rotary motor with 8000-line analog encoder and 3 pole pairs should set CA[18]= 8000x2^{16} = 524288000 and CA[19]=3

Note: For good commutation, the number of feedback counts per electrical rotation (CA[18]/CA[19]) should be at least 36.

4.2.3 Initializing the High Resolution Position Sensor Manually

The high resolution position counter can be set manually by the TW[63] command. The use of TW[63] is here clarified by an example. Suppose that a motor has 3 pole pairs (i.e., 12 steps) and an 8000 count/rev incremental encoder. We shall program CA[18]=8000, CA[19]=3, and one electrical cycle of the motor will count approximately 8000/3=2667 encoder counts.

If we set TW[63]=1000, we assert that the present electrical position of the motor is (1000/2667)×360° which is approximately 135°.

Setting TW[63]=3667 is similar, since mod(3667,2667)=1000. In other words, the value we set into the commutation counter reflects a full electrical cycle of 2667 counts, plus the 1000 counts that matter.

Note:

TW[N] commands are privileged; they work only after applying TP[6]=1.

4.3. Auto-phasing and Commutation Search

Auto-phasing is the process of initially estimating the rotor electrical angle, without any Hall or absolute sensor.

SimpIIQ supports the following methods for commutation initialization:

- Oscillation. This method is good when the motor may move a little on start.
- 2nd harmonics. This method is good when the motor is braked before start. Its success depends on the magnetic structure of the motor.

The parameter CA[17] selects the auto-phasing method to use.

4.3.1 Auto Phasing by Oscillation

When starting the motor, the rotor can be located anywhere. The torque of a brushless or stepper motor is given by the equation:

$$T = K_T I \sin(\theta) \quad (1)$$

$$\theta = \theta_s - \theta_r \quad (2)$$

where:

T is the motor torque.
K_T is the motor constant.
I is the motor current.
θ is the electrical angle between the rotor and the field at the stator.
θ_s is the electrical angle of the stator field.
θ_r is the electrical angle of the rotor.

The angle θ_S is known because the drive controls it directly. The angle θ_r is unknown.

If θ_S is rotated — that is, θ_S –$2\pi f * t$, where *f* is a frequency and *t* is the time — the result is the following sinusoidal torque:

$$T = K_T * I * \sin(2\pi f * t - \theta_r) \quad (3)$$

For that torque, the motor shaft will move according to:

$$P(t) = A(f) * K_T * I * \sin(2\pi f * t - \theta_r - \phi(f)),$$

where:

P(t) is the motor shaft position.

$A(f)e^{j\phi(f)}$ is the transfer function of the motor and its load at the frequency *f*.

By applying a torque of (3) and measuring the position, both A(f) and θ_r can be identified.

4.3.1.1 Selecting Parameters

The parameters I and f can be selected from the following range:

Parameter	Description
I	The torque is selected by the parameter CA[26], which defines I as a percentage of the continuous current rating CL[1]. For example, if CA[26] = 50, then I = 0.5 CL[1].
f	CA[15] controls the frequency of the sinusoidal motor torque. The basic frequency (with CA[15] = 0) is such that a cycle is completed in 128 * TS microseconds. For a *SimpIIQ* drive with TT[1]=40, this will be a cycle of 5120 microseconds, which is about 200 Hz. The frequency is $2^{CA[15]} *$ basic frequency of CA[15] in the range [-4…4]. For example, with CA[15] = 2, the frequency is about 80 Hz, whereas with CA[15] = -2, the frequency is about 1280 Hz.

The selection rules for parameters I and f are as follows:

- The torque I must be as large as possible so as to reduce the relative effect of disturbance torques (such as cogging and friction) on the resulting waveform. Normally, I is taken as about 50% of the continuous motor current.
- The frequency f must be selected so that the amplitude of the position sine is at least 1 electrical degree, and at least 6 to 8 bits. The motor position can be analyzed accurately enough when the position sine amplitude is 4 encoder bits or more. A slightly higher amplitude is set to prevent minor load changes from disturbing the analysis. Larger position amplitudes will work, but the shaft oscillation at the motor starting process will be unnecessarily large.
- The frequency f must be such that in that frequency, the load behaves inertially. This means that in that frequency, the phase angle ϕ(f) is in the range of [-140…-220] degrees, and that the amplitude function A(f) does not have the high gradient of near resonance regions. The algorithm will not function near a resonant frequency, or in a low frequency in which a large viscous friction is present.

- Specifying a very high oscillation frequency (CA[15] = -3 or -4) is not recommended, because not enough position samples will be available for each sine cycle.

The setup program normally selects the parameters I and f at the "Establishing Commutation" stage of auto-tuning.

4.3.1.2 Method Limitation

The algorithm presented here is quite robust, and should work for most motor systems, including those with moderate backlash. It does, however, have the following limitations:

- The encoder must have a resolution of at least 256 counts per pole pair. For example, if a motor has three pole pairs, 256 lines (1000 counters/revolution) will suffice. An encoder with 128 lines will not.
- The system cannot be extremely unbalanced. It is essential that the motor does not accelerate significantly when no current is applied.
- The motor cannot be free to oscillate in both directions.
- The motor static load should not change significantly. If the static load is subject to considerable changes, tune the algorithm with the highest possible load. Namely, the inertia of the load must be above 40% of its value from when the parameters were tuned, and it cannot exceed 40% of its value from when the parameters were tuned. If the load inertia is too high, the present torque level will not suffice for oscillating the motor shaft. The algorithm will fail and the motor will not start.
- The algorithm assumes that in the excitation frequency, the motor and the load behave like inertia. This assumption fails in the following cases:
  - The excitation frequency is a resonant frequency of the system.
  - There is a large viscous (speed-dependent) friction and the excitation frequency is so low that the phase of the transfer function between the torque and acceleration is not in the range of [-30…30] degrees. If the transfer function phase is out of range, the algorithm will fail and the motor will not start.

4.3.1.3 Protections

The method described here can work reliably in many practical situations, although it does not match every application. If the parameters of the method are not tuned properly, or if the method does not match the application, the motor starting process will fail.

After setting MO=1, the algorithm will attempt to rotate the motor back and forth and find the commutation angle. If the results are not reliable, because the oscillation amplitude is too small or because the load behavior is not inertial, MO will reset automatically to zero and the motor will not start.

4.3.1.4 Maximum Number of Iterations for Auto-phasing

If the auto-phasing algorithm fails (the motion amplitude is less than CA[24]), it may retry with the same frequency but with a doubled current.

The CA[27] parameter defines the maximum number of retries for the auto-phasing process. If the process fails due to overload, the auto-phasing repeats, doubling the current every iteration. If the peak current limit is reached during an attempt, the auto-phasing process will stop.

4.3.1.5 Starting the Motor without Digital Hall Sensors

The following parameters are relevant to starting the motor when digital Hall sensors are not used.

Parameter	Description
CA[18] CA[19]	Encoder resolution and number of motor pole pairs. CA[18] must be greater than CA[19] * 256.
CA[20]	Must be set to 0 to indicate that no Hall sensors are present.
CA[21]	Must be set to 1 to indicate that an encoder is present.
CA[15]	Set the oscillation frequency as explained previously.
CA[24]	Set the minimum acceptable oscillation amplitude to 4.
CA[26]	Set the excitation amplitude as a percentage of the continuous current.
CA[27]	Set the maximum number of iterations for the auto-phasing process.
MO	MO=1 sets the motor to on. If the motor starts to fail, MO resets to 0.
WS[15]	Reads the actual oscillation amplitude.
WS[16]	Flag if a auto –phasing failed because of suspected resonant frequency near the excitation frequency.

Chapter 5: Limits, Protections, Faults and Diagnosis

This chapter describes the limits and protections implemented by *SimpIIQ* drives. Limits are software restrictions that prevent *SimpIIQ* drives from running into dangerous situations. For example:

- Limits set for the torque command prevent the motor or *SimpIIQ* drive from burning.
- Reaction to limit switches stops the motor before it accidentally hits something out of its expected motion range.

Protections prevent the motor from starting, or shut down an active motor due to abnormal situations. For example:

- Bad or inconsistent setup data prevents the motor from starting.
- An unexpectedly high motor current can endanger the *SimpIIQ* drive and the motor. Such a high current is abnormal because the command to the current (torque) amplifier is limited. If the current controller seems to be functioning poorly, the drive is immediately shut down.
- The drive is too hot.

When the drive shuts down due to a protection, the motor continues to run on its own inertia unless brakes are used. In order to avoid spurious motor shutdowns, always:

- Leave enough space between the limits and the protection. For example, HL[2] specifies the over-speed limit. A protection is activated when VX > HL[2]. VH[2] specifies the legal command to the speed controller. Speed commands over VH[2] are clipped to VH[2]. The protection HL[2] must be greater than the limit VL[2]. Moreover, the distance HL[2] -VH[2] must leave enough space for the expected speed overshoot.
- When safety is a concern, always use brakes. *SimpIIQ* drives can program one of their digital outputs to activate a brake immediately upon motor shutdown.

If an exception stops the motor or prevents the motor from starting, the *SimpIIQ* drive, in most cases, can identify the cause of the event. If a real-time exception reaction is desired, a #@AUTO_ER routine can be added to the user program

The following commands are used with limits and protections:

Command	Description
BP[N]	Brake parameters.
CD	Dumps the process status of the *SimpIIQ* drive and reports database inconsistencies.
CL[N]	Continuous current limit and “Motor not moving” protection.
EC	Error code describing why previous command returned an error.
ER[N]	Tracking error exception limits for speed and position.
HL[N]	Protection high limits for position and speed feedback.

Command	Description
IL[N]	Input logic. Defines digital inputs as stop and limit switches.
LC	Limit current indication that indicates whether peak limit is active.
LL[N]	Protection low limits for position and speed feedback.
MF	States reason for a motor automatic shutdown.
OL[N]	Output logic. Programs digital outputs as brake activation or as drive ready indication.
PL[N]	Peak current limit.
PS	Program status.
SR	Status register. Gives *SimpIIQ* drive status, including if motor is shut down by exception or if a Stop switch has been hit.
VH[N]	Command high limits for position and speed reference.
VL[N]	Command low limits for position and speed reference.

Figure 5-1: **Commands Relevant to Limits, Protection and Diagnosis**

CANopen offers an additional protection level: emergency objects issued in cases of errors or applied protections (refer to the Elmo *CANopen Implementation Manual)*.

5.1. Current Limiting

The drive protects the motor from over-current using a two-stage method. The motor maximum peak current (given by PL[1]) is available for the peak duration time (specified by PL[2]), while for longer periods, the current is limited to its continuous limit (CL[1]).

The current limiting process is dynamic: If the current has been close to its continuous limit, the time allowed for the peak current is reduced.

The following block diagram shows the current limiting mechanism:

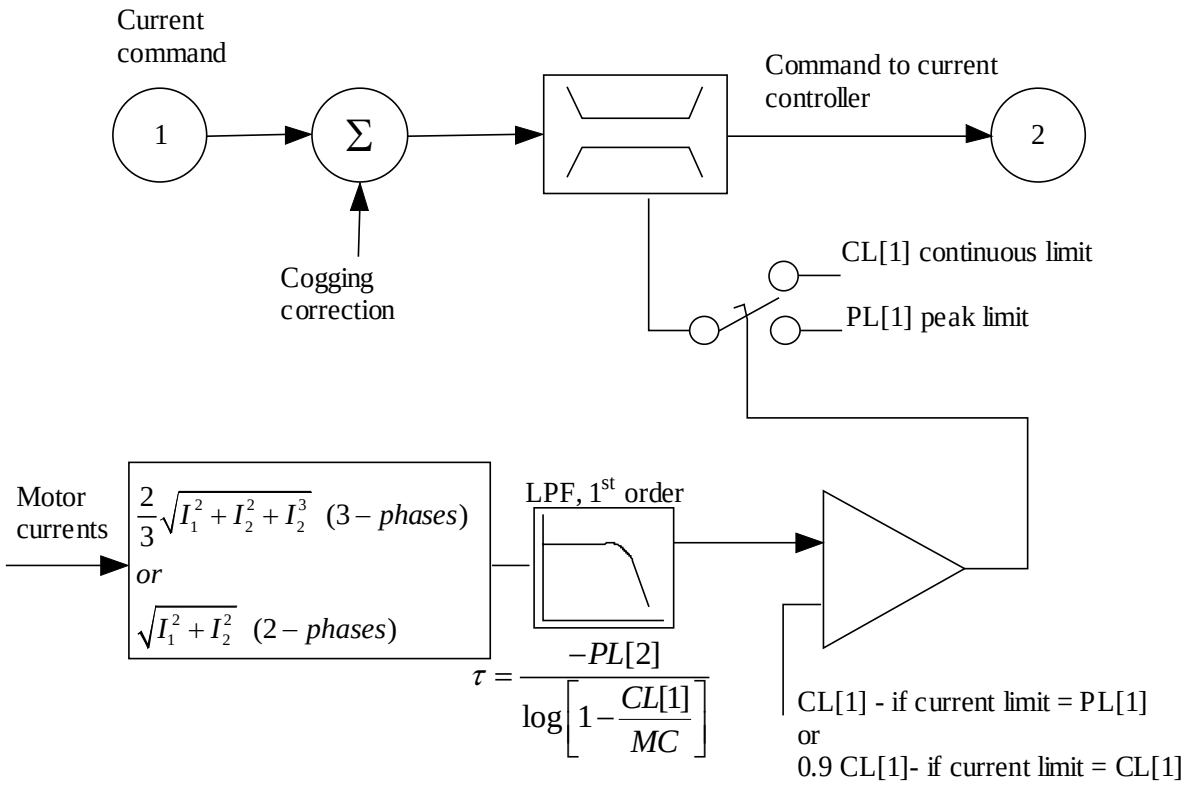


Figure 5-2: **Current limiting mechanism**

In the figure, we see that the RMS currents are averaged using a low-pass filter. When the output of the filter crosses the threshold of CL[1], the current limit drops to CL[1]. It will remain so until the averaged RMS current will fall below $0.9\times CL[1]$

The time constant of the averaging low-pass filter is taken as the minimum between $\frac{3\text{sec}}{\left[1-\frac{CL[1]}{MC}\right]}$ and $\frac{PL[2]}{-\log\left[1-\frac{CL[1]}{PL[1]}\right]}$, where MC is the maximum servo drive current, and PL[2] programs the peak current maximum duration.

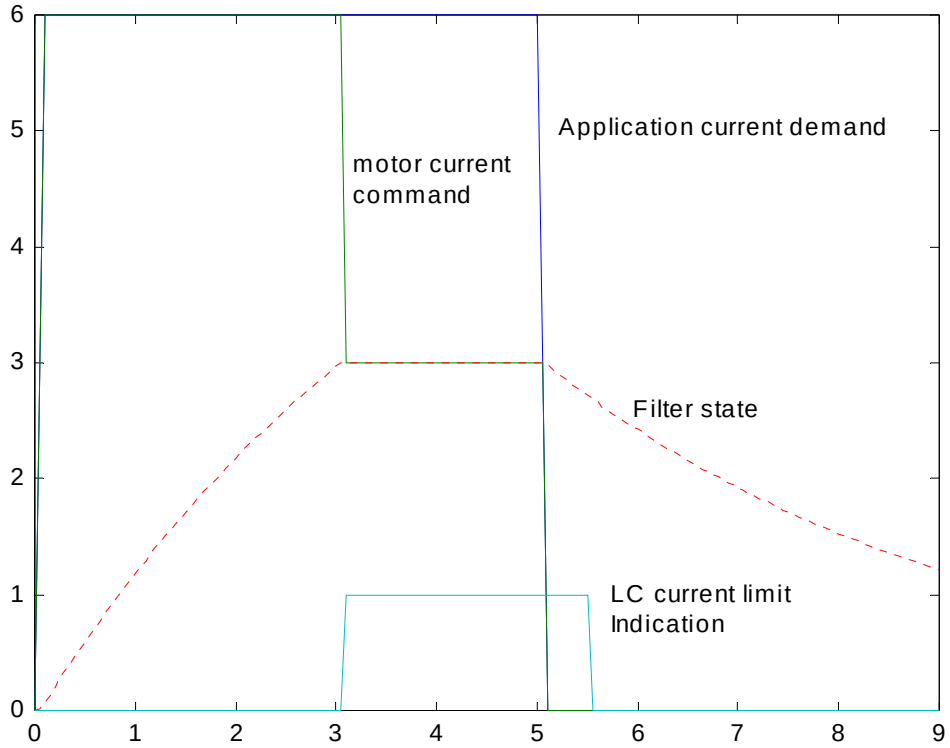
The value of τ is so that when a current demand of PL[1] is placed after a long period of zero current, the current demand may be satisfied for PL[2] seconds.

For value of PL[2] close to MC, however, the peak current time is limited to 3 seconds.

Notes:

- If, prior to the high current demand, the current demand was very close to CL[1], the switch will occur almost instantaneously.
- If the current demand is just greater than CL[1], and significantly less than PL[1], the switch to current limiting may take a very long time.

Example:
The following graph depicts the signals related to the current command limiting process for MC=6, PL[1]=6, PL[2]=3 and CL[1]=3.



The application motor current command increases from 0 to 6 amperes at the time of 0 seconds, and then decreases to 0 at the time of 5 seconds.

The state of the filter (averaged RMS current) increases until it reaches the continuous current limit of 3Amp at the time of 3 seconds. At that time, the LC flag is raised and the motor current command decreases to 3 Amperes.

After the motor current command is set to zero, the state of the filter begins to drop. When it drops to 2.7 amperes = 90% of 3 amperes, the LC flag is reset and the torque command limiting returns to 6 amperes.

5.2. Sensor Faults

5.2.1 Motor Does Not Move Protection

When the motor is unable to complete a command to move, the reasons may be:

- The motion sensor is faulty: The motor moves but motion is not detected. In this case, AC motors will generally stop, because the stator field will remain stationary.
- The motor is faulty or another mechanical failure is preventing the motor from moving.

- The controller filter is poorly tuned. In this case, the motor torque may oscillate wildly at high frequency, but the motor will barely move.

Indications of such situations include:

- High average motor torque
- Stationary or hardly moving motor

A stationary motor responding to a high torque command does not always indicate an error. In certain applications, such as thread fastening, it is perfectly legitimate for the motor to reach a mechanical motion limit.

The drive user should define whether a high-torque stopped motor is a fault or not. If the parameter CL[2] is less than 2, a high torque that does not lead to motion is not considered a fault. If the parameter CL[2] is 2 or more, a high-torque stopped motor, detected for at least 3 continuous seconds, is considered a fault. The motor is set to off (MO=0) and the fault is MF=0x200,000. The time constant of 3 seconds is used because almost every motion system applies high torques for short acceleration periods while the speed is slow.

CL[2] defines the tested torque level as a percentage of the continuous current limit CL[1]. CL[3] states the absolute threshold main sensor speed under which the motor is considered not moving. CL[3] should not be set to a very small number because when a motor is stuck, a vibration may develop so there will be non-zero speed-reading. When an encoder wire is damaged, the encoder readout is vibrating $\pm$ bit. This also creates speed-reading.

Example:
If `CL[2]=50` and `CL[3]=500`, the drive will abort (automatic MO=0) if the torque level is kept at at least 50% of the continuous current, while the shaft speed did not exceed 500 counts/second for a continuous 3 seconds.

5.2.2 Commutation Faults

5.2.2.1 Matching the Hall Sensors with an Incremental Position Sensor

If both an incremental sensor and digital Hall sensor are available for commutation (CA[17]=1), then *SimpIIQ* will continuously check that both the sensors are consistent.

The Hall sensors provide a rough, but direct, estimate of the commutation angle. The encoder provides high-resolution data, but it may accumulate errors if the drive setup data is set incorrectly or if it is faulty.

If the commutation angle derived from the encoder is more than 15 electrical degrees out of the range of Hall sensor reading[7], *SimpIIQ* increments an error counter. If the readings match, the error counter is reduced.

[7] Digital Hall sensors resolution is only 60 electrical degrees.

If the error counter reaches the threshold of PN[1], The motor is shut down (MO=0) and MF is set to 0x4.

After detection of a commutation fault, the motor must be shut down and a proper MF code set to be active. The motor cannot be stopped under control, because the correlation between the motor currents and the resulting torque is not known. At the next motor startup attempt, the motor will again seek commutation.

Chapter 6: The Position Reference Generator

The position reference signal is generated by the following components

- Software reference generator
- External position reference generator
- Stop manager

The contents of this chapter apply to:

- Closed loop position control (UM=4, UM=5)
- Open loop stepper control (UM=3)

6.1. Software Reference Generator

The *SimpIIQ* drive supports five modes of software referencing:

- Idle: motor standing in place
  This is the default mode after setting motor on, and after an interpolated motion is complete.
- PTP: point-to-point
  The user specifies the position in which the motor is to stand, and the limits subject to which the trajectory to the target should be designed. The *SimpIIQ* drive automatically designs a minimum-time trajectory and executes it.
- Jog
  The user specifies the speed at which the motor should steadily move and the acceleration limits to implement towards that speed. The *SimpIIQ* drive automatically designs a minimum-time trajectory and executes it.
- PT: position/time
  The user specifies a set of points to be visited at fixed points of time. The *SimpIIQ* drive interpolates a third-order polynomial between the user's points. The speeds at the user points are selected in order to form a smooth motion trajectory. Using PT, complex motions are easily designed. PT data may be transferred to the drive online using an efficient CAN PDO protocol, so that infinite PT motions are possible.
- PVT: position/velocity/time
  The user specifies a set of motion points, each of which includes a position and the speed and time at which the position should be visited. The user points are interpolated using third-order polynomials. PVT mode allows absolute time specification so that several drives may compose a fully-synchronized motion. PVT data can be transferred to the drive online using an efficient CAN PDO protocol, so that infinite PVT motions are possible.

In order to initiate a software motion, its parameters must be initialized and a mode command issued. The next BG (software, hardware or timed) will start the motion. The following are the mode commands:

Command	Mode
ST	Idle: ST stops any motion.
PA	PTP: PA=n specifies a PTP motion, to absolute position n counts.
JV	Jog: JV=n specifies a jog motion, at speed n counts/second.
PT	PT: PT=n specifies a PT motion, beginning from the nth item in the PT data table.
PV	PVT: PV=n specifies a PVT motion, beginning from the nth item in the PVT data table.

Table 6-1: **Software Motion Mode Commands**

6.1.1 Switching Between Motion Modes

The ST command can be used for stopping at any time. The drive will decelerate the position or velocity reference until it reaches a full stop.

PTP and jog motion commands can be sent anywhere, at any time. The drive will calculate the path to be followed so that the desired speed or position is reached, subject to the acceleration limits. PTP and jog motions can even be initiated during PVT or PT motion. Care must be taken, however, if the smooth factor (SF) is nonzero. Upon switching from interpolated motion to PTP or jogging, the PTP or jogging will start unsmoothed, with smoothing gradually building up, until after SF milliseconds, the motion is fully smoothed.

Care is also required when switching to tabulated (PT and PVT) motion modes. These modes should be started only when the motor is at a complete stop (MS=0 or MS=1), at the exact position specified as the first point for the PT or PVT. If the motor is not totally stopped at the correct start position, the software reference will jump. The command to the position controller will attempt to catch up with the PT or PVT motion, using the SD acceleration.

6.1.2 Comparing PT and PVT Interpolated Modes

The following tables compare PT and PVT motion modes.

Feature	PT	PVT
Motion buffer length	1024	64
Position points in one CAN message	2	1
Speed command calculation	Automatic, by interpolation	Specified by user
Acceleration command	Automatic, by interpolation	Automatic, by interpolation
Time between user data points	Fixed multiple of the 1 - 255 controller sampling times. Cannot be changed on-the-fly.	Specified by user independently for each motion interval, in msec. Range: [1…255] msec.
Cyclical motion support	Yes	Yes

Feature	PT	PVT
On-the-fly motion programming with handshake host protocol	Yes	Yes

Table 6-2: **Tabulated Motion Differences**

Feature	Preferred
Long preprogrammed motions	PT
Ease of use	PT
Variable command sampling time	PVT
Motion design independent of controller sampling time	PVT
Synchronized, multiple-axis motions	PVT

Table 6-3: **Tabulated Feature Preferences**

6.1.3 Idle Mode and Motion Status

After motor on, the position profiler is idle, meaning that the software position command is fixed. The idle mode is revisited whenever a mode terminates (a PTP or tabulated motion is completed) or a stop (ST) command is issued.

Idle mode is the only mode in the parameters of the external reference generator can be changed. This mode can be identified by the MS (motion status) variable, which can be read as follows:

MS Value	Description
0	Position reference generator is idle and motor position is stabilized within target radius for a satisfactory length of time.
1	Position reference generator is idle or the motor is off (MO=0).
2	Position reference generator is active in one of the optional motion profilers: PTP, jog, PT or PVT.

Table 6-4: **Motion Status Indications**

When MS is 1 or less, the parameters of the external reference generator can be switched. The condition MS=0 implies that the motor position is indeed stabilized, meaning that the motor position is within the target radius, for at least the target time. The target radius is given by RS[1] counts, and the target time is TR[2] milliseconds.

Example:
The concepts of target time and target radius are demonstrated in the following figure.

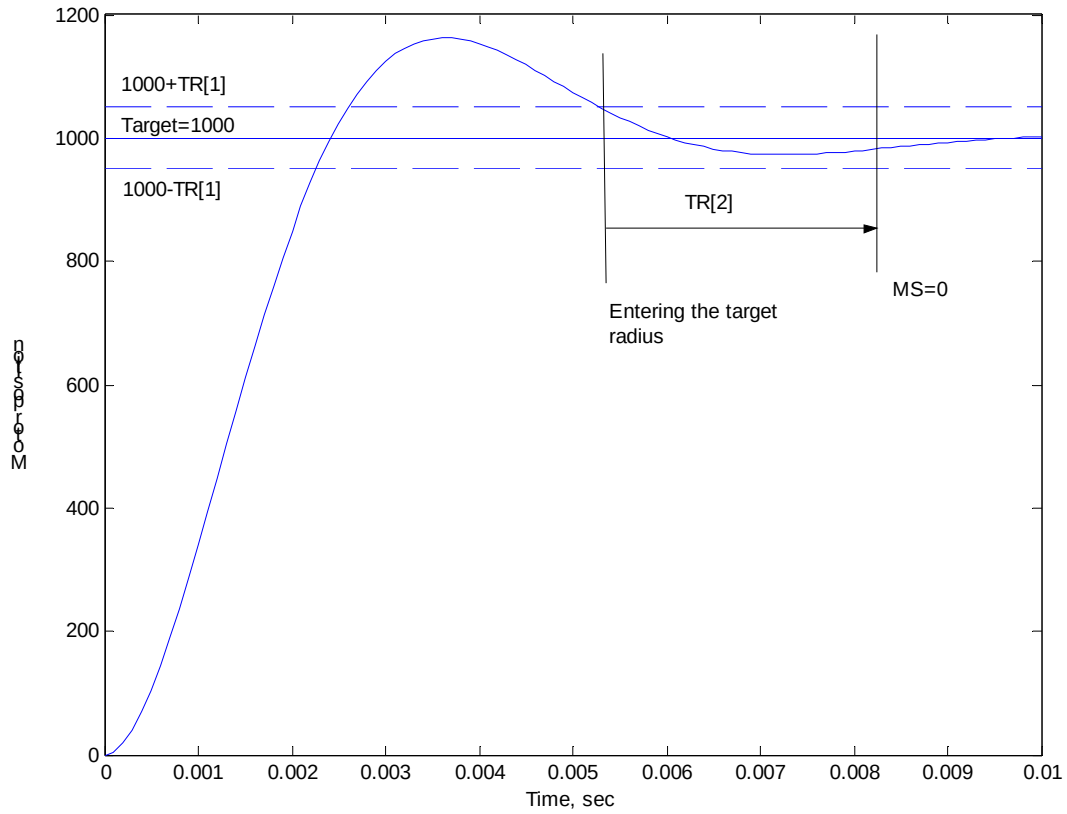


Figure 6-1: **Target Time and Target Radius**

In this figure, the motor position settles after overshooting the target of 1000 counts. The target radius is TR[1]=50.

The drive sets MS=0 (converged to final position) after the motor position is within the target radius for long enough time.

Note that while overshooting, the motor position was briefly within the target radius; the motion status (MS) ignore this brief crossing as it did not sustain for TR[2] milliseconds.

6.1.4 Point-to-Point (PTP)

6.1.4.1 Basic PTP

In this motion mode, the motor moves from its present position to a final point at zero speed, and stays there. The trajectory to the final point is calculated based on speed, acceleration and deceleration limits, as set by the AC, DC and SP parameters respectively.

The largest PTP motion available is (XM[2] - XM[1])/2, because with modulo calculation the PTP motion always goes the short way around. For example, if XM[1]=-500 and XM[2]=500, the present position reference is 490 and the command PA= -490. When BG is entered, the position reference increases and goes through 499 to -500, and then to -490. The total length of the movement will be 20 counts.

The parameters of PTP motion are summarized in the following table:

Parameter	Action
AC	Acceleration, in counts/second^2
DC	Deceleration, in counts/second^2
SP	Maximum speed, in counts/second
SF	Smooth factor, in milliseconds
PR	Relative position, in counts
PA	Specifies that the next motion will be PTP, and the absolute target position

Table 6-5: **PTP Motion Parameters**

The PA command plays more than one role. PA=n specifies that the next BG will start a PTP motion. In addition, it specifies the target of the next PTP move.

In PTP mode, a BG command performs the following:
`PA=PA+PR`
`Go to PA`
where:
`PA` is the absolute position target.
`PR` is the relative position target, enabling the specification of an incremental move or a series of incremental moves. `PR` is reset to zero by the `PA=n` command. Therefore, `PA=n; BG` initiates a motion towards `n`.

The value of PA is incremented automatically with PR every time a new motion is initiated. For example:
`PA=0; BG; while(MS==0); PR=1000; BG; while(MS==0); BG; while(MS==0); BG`
initiates four consecutive PTP motions, targeted at 0, 1000, 2000 and 3000 counts respectively.

PTP example:
The simplest point-to-point motion is from a stationary position to another stationary position. The acceleration and final speed limits are:
`AC=100,000, DC=200,000, SP=2000,`
starting from `PX=0 and PA=70; BG;`

The following speed graph shows the acceleration, constant speed and deceleration in the trajectory to the target.

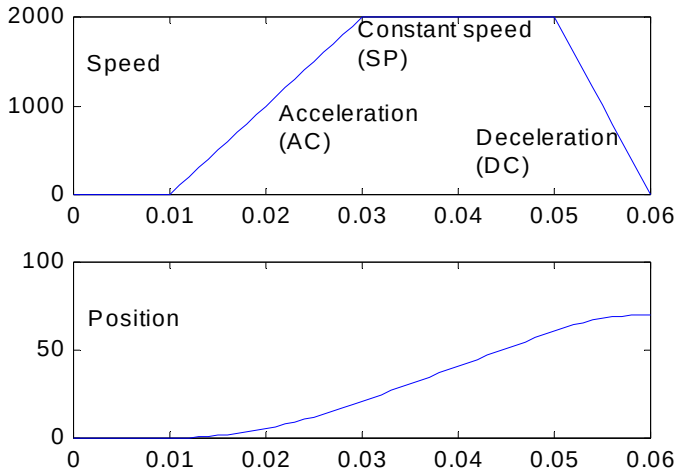


Figure 6-2: **Trapezoidal speed profile**

With shorter movement, the deceleration begins before the speed limit is reached, so that the SP speed limit is not effective. This situation is depicted in the following figure:

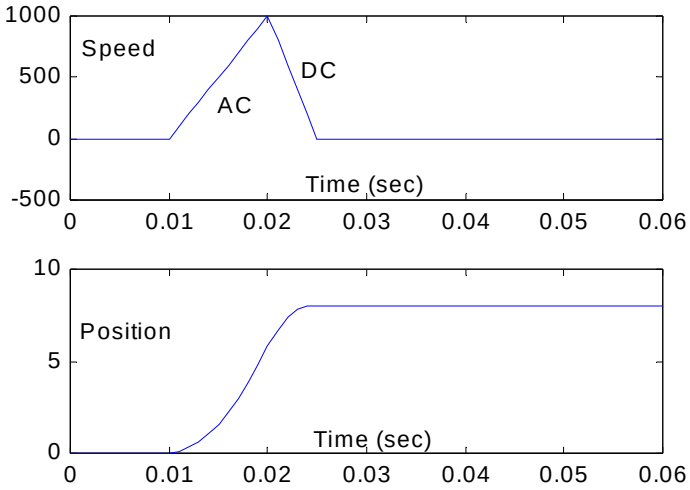


Figure 6-3: **Triangular speed profile**

6.1.4.2 More Complex PTP Motions

PTP motions may be initiated any time, using the PA command, but not necessarily from a stationary state. The PTP decisions, made every position control cycle, are described in the following flowchart. All parameters in the flowchart — including AC, DC, SP and position target — are updated by a BG command, or by its hardware-activated equivalent.

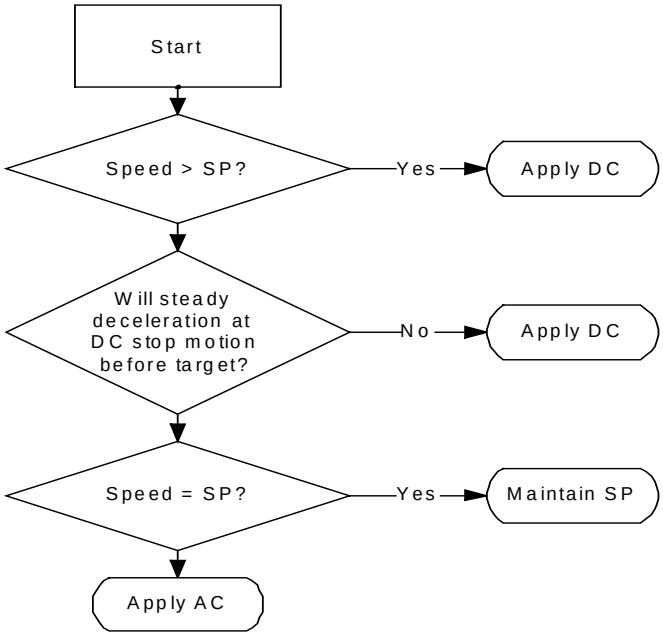
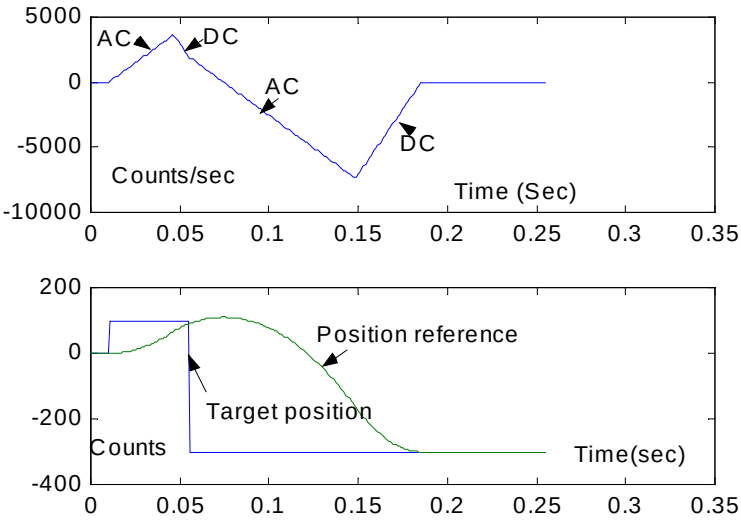


Figure 6-4: **PTP Decisions Flowchart**

Example of changing the position target on-the-fly:
The acceleration and final speed are:
`AC=100,000, DC=200,000, SP=20,000,`
starting from
`PX=0` and target position `PA=100; BG;`
The position reference has not yet stabilized at the position target, when the position target is changed by the command `PR=-400; BG;` to the position target of `(PA+PR)=-300`. The drive calculates the position reference to reach the new target.

Note that in this example, the position reference overshoots the original position target of 100. This is because at the time that the position target was changed, the drive was approaching the target using DC. When a new target is set, the drive exits the state of final approach. Therefore, it uses the AC acceleration, which is smaller than DC in this example. With the reduced acceleration, it cannot avoid the overshoot.



6.1.5 Jog

In a jogging motion, the motor receives a command to move at a fixed speed. The AC and DC parameters indicate the acceleration or deceleration to the desire speed.

Jog motions may be initiated any time by using the JV command, and not necessarily from a stationary state. The jog mode decisions, made every position control cycle, are given in the following flowchart.

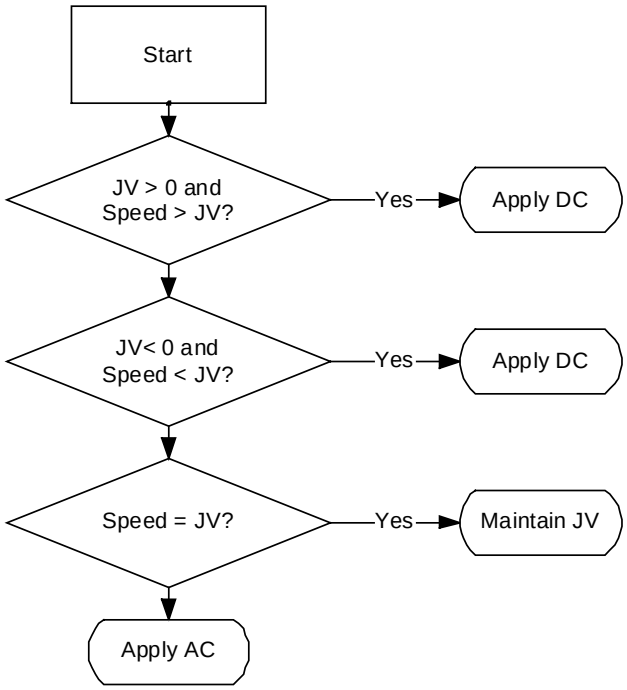


Figure 6-5: **Jog Decisions Flowchart**

All parameters in the flowchart — including AC, DC, JV and the position target — are updated by a BG command or by its hardware equivalent (refer to the IL[N] command in the *SimpIIQ for Steppers Command Reference Manual).*

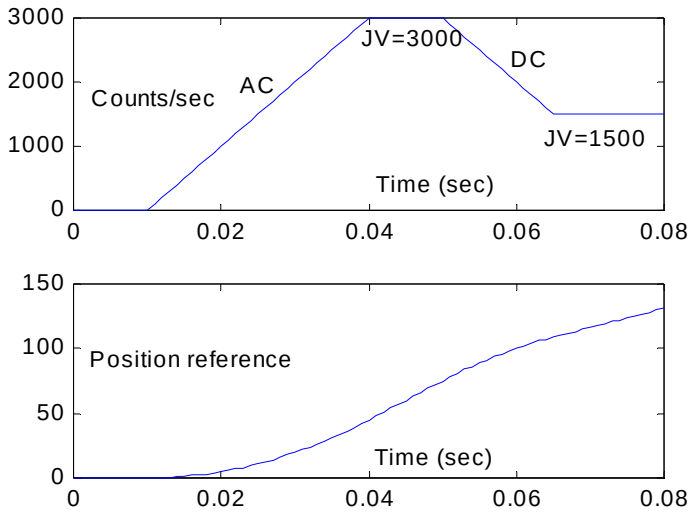
Jog motions can continue indefinitely. The position reference jumps by XM[2] - XM[1] when it reaches the modulo boundary, but the speed remains constant. The jog motion parameters are summarized in the following table:

Parameter	Action
AC	Acceleration, in counts/second^2
DC	Deceleration, in counts/second^2
SF	Smooth factor, in milliseconds
JV	Jog velocity

Table 6-6: **Jog Motion Parameters**

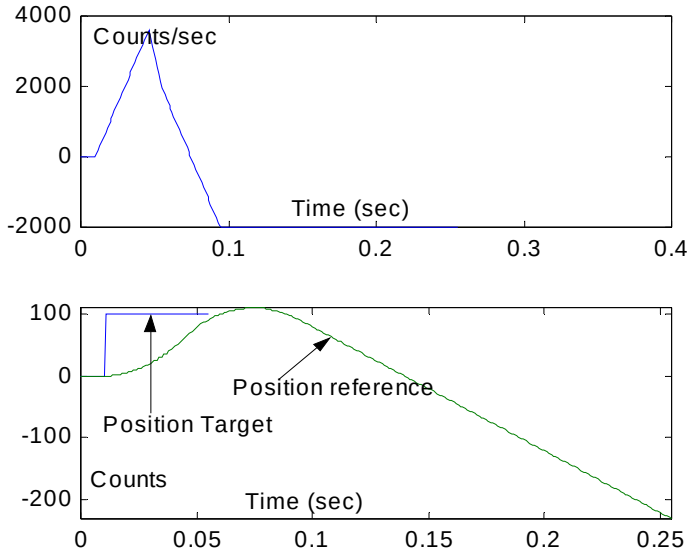
The JV command specifies that the next motion will be a jog, and also what speed the jog will target.

Example of simple jogging:
Begin with the command sequence `JV=3000; BG`
The position reference starts to accelerate until the jog speed reaches 3000. Later, the command sequence `JV=1500; BG` changes the speed of the position reference to 1500 counts/second.



Example of on-the-fly mode switching:
Begin with the command sequence: `AC=100,000; DC=200,000; PA=100; BG;`

A PTP motion begins, during which the drive brings the reference position to the position target. At the time of 0.055 seconds (where the plot of the position target terminates in the following figure), the command `JV=-2000; BG;` is entered. The drive uses the AC acceleration to reach the desired speed. If, at the time of the `BG`, the speed of the position reference is more negative than -2000 the drive will use the DC parameter instead of AC.



6.1.6 Position - Velocity - Time (PVT)

In a PVT motion the user provides the desired position and speed at selected time instances. Between these specified times, the motion controller interpolates to obtain smooth motion. The position and speed specifications are absolute, while the time specification is relative.

A PVT motion can be referenced in absolute time by requiring it to start at a specified time. The BT (Begin on Time) command is used to start a PVT motion exactly at a given time, using a microsecond resolution.

The ability to relate PVT motions to absolute time makes them ideal for tightly-synchronized multiple-axis motions. For an explanation of how to synchronize the absolute time counters of multiple drives to the precision of a few microseconds, refer to the Elmo *CAN Implementation Manual.*

Example 1:
The system should be at position 1000 and speed 100,000 counts/second at a given time, and at position 1620 and speed 110,000 counts/seconds 6 milliseconds later.

The first point is specified by:
Position = 1000,
Speed - 100,000,
Time = (irrelevant to described motion, but relevant to the previous motion segment)

The second point is specified by:
Position = 1620,
Speed = 110,000,
Time = 6 msec

The time is defined in milliseconds, not in drive sampling times. The drive interpolates the motion specification in order to calculate the desired position and speed at the sampling instances, when it needs the information.

PVT implements a third-order interpolation between the position - speed data provided by the user. For each motion interval, the user specifies the boundary positions and speeds. Mathematically, the user provides the following data:

- Starting position and speed, denoted by P0 and V0, respectively
- End position and speed, denoted by PT and VT, respectively

If t_0 denotes the starting time and T denotes the length of the time interval,
the position for $t \in [t_0, t_0 + T]$ is given by the third-order interpolating polynomial:
$P(t) = a(t - t_0)^3 + b(t - t_0)^2 + c(t - t_0) + c$

The speed is given by:
$V(t) = 3a(t - t_0)^2 + 2b(t - t_0) + c$

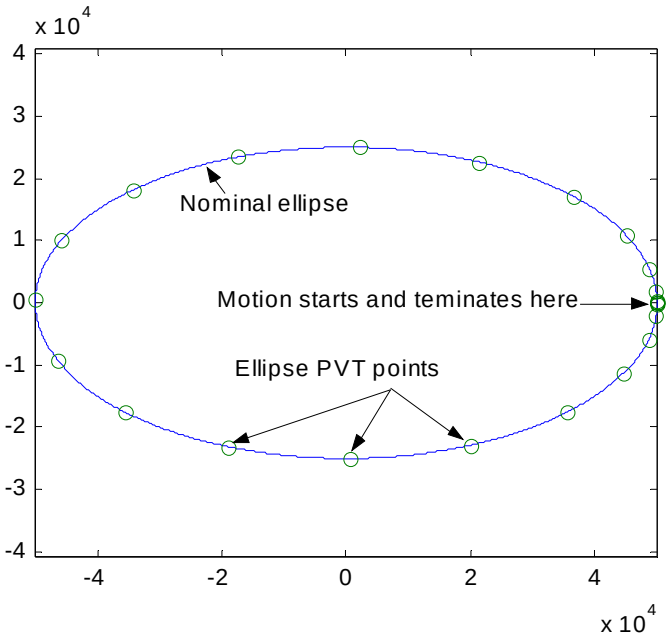
The four parameters a, b, c and d are unknown and can be solved using the following four linear equations:
$P(t_0) = P0$, namely, $d = P0$
$V(t_0) = V0$, namely, $c = V0$
$P(t_0 + T) = PT$, namely, $PT = aT^3 + bT^2 + cT + d$
$V(t_0 + T) = VT$, namely, $VT = 3aT^2 + 2bT + c$

Example 2:
This example demonstrates how a very few points can accurately describe a smooth and long motion path.

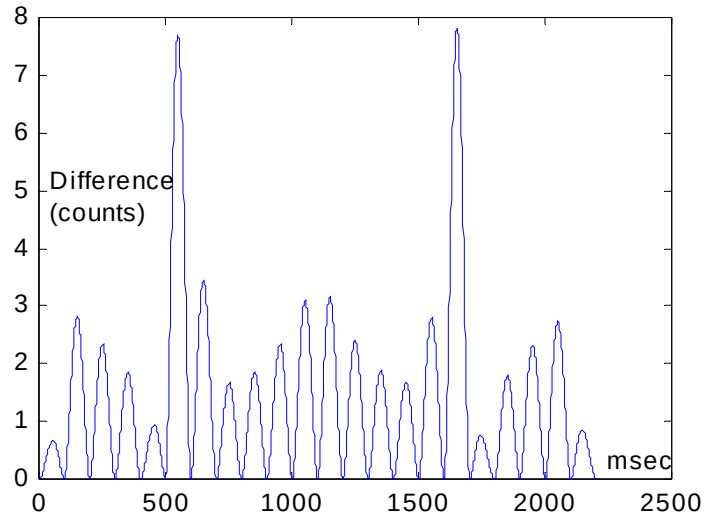
Two drives, driven synchronously, draw an ellipse. One drive drives the x-axis and the other drives the y-axis. The long axis of the ellipse is 100,000 counts long and the short axis of the ellipse is 50,000 counts long. The entire ellipse is to be traveled within 2.2 seconds.

A PVT motion is planned with a fixed inter-point time of 100 milliseconds. For the entire ellipse, 23 points are sufficient, as depicted in the following figures. The motion is planned so that the tangential speed is accelerated to a constant rate, and then decelerated back to zero at the end of the ellipse. Near the starting point of the ellipse, the speed is slow; therefore, the PVT points — which are equally spaced in time — are more spatially dense in that area. The continuous line in the following figure depicts the true ellipse along with the ellipse generated by the drive by interpolating the PVT points.

The original ellipse and the drive interpolation of the PVT points are so close that they cannot be differentiated on the plot, as seen in the following depiction of the ellipse.

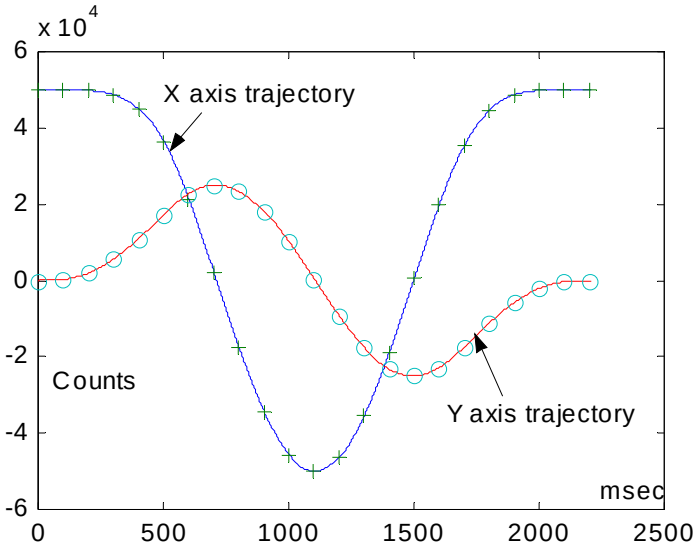


The following figure takes a closer look at the error between the true ellipse and the drive-interpolated path.

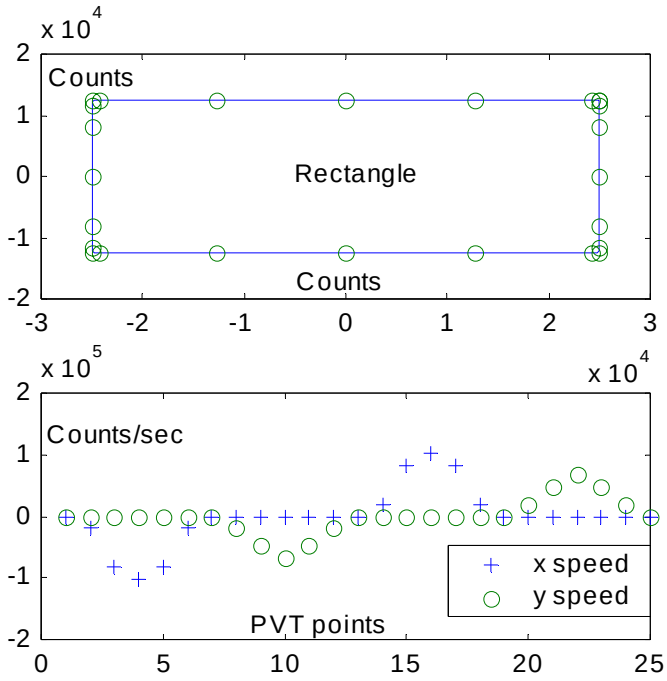


It can be seen that with only 23 PVT points, the interpolated path never differs from the original ellipse by more than 8 counts (remember that the ellipse axes are 100,000 and 50,000 counts respectively!) At the PVT points, the interpolation error is zero.

The next figure displays the interpolated trajectories generated by the drive for the x-axis and for the y-axis.



Example 3:
The drive normally produces maximally smooth interpolating trajectories, which is why the ellipse in the previous example could be interpolated using so few points. Accurate *corners*, however, require non-smooth interpolation. Specifying zero speed for the corner points generates hard corners. The following graph shows a 2-axis synchronized PVT trajectory of an accurate rectangle.

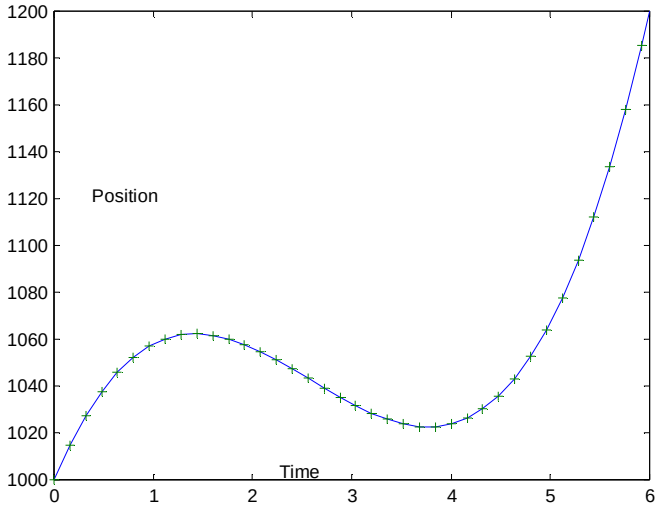


For the corner points both the x and y speeds are specified to zero. The interpolation error for the entire rectangle is zero.

Example 4:
A PVT interpolation interval contains these values:
Starting position = 1000 counts
Starting speed = 100,000 counts/second
End position = 1200 counts
End speed = 190,000 counts/second
Time = 6 milliseconds

The interpolated path for the data of the table, for a controller with a sampling time of 160 milliseconds, is depicted in the following figure. The + symbols indicate the points at integer multiples of the controller sampling time. At these points, the drive evaluates the interpolated motion path.

The end point does not necessarily fall on a controller sampling time. Strange-looking position commands may result if the speed choice is not consistent with the position and time definitions. The position distance and the time between points imply an average speed of (1200 - 1000)/0.006 = 33,000 counts/second. This average conflicts with the boundary point speed specifications of 100,000 counts/second and 190,000 counts/sec, respectively.



6.1.6.1 The PVT Table

A three-column table is used to define PVT motion. Each row of the table defines the position and speed at a single time instance.

#Index	P (32 Bits)	V (24 Bits)	T (8 Bits)
1	QP[1]	QV[1]	QT[1]
2	QP[2]	QV[2]	QT[2]
…	QP…	QV…	QT…
64	QP[64]	QV[64]	QT[64]

Table 6-7: **PVT Table**

The table has 64 rows, enabling the specification of up to 63 consecutive PVT motion segments (64 segments if the table is used cyclically). The cells of the table may be accessed using the QP, QV and QT commands:

- The QP[N] command sets/reads the nth row of the P column.
- The QV[N] command sets/reads the nth row of the V column.
- The QT[N] command sets/reads the nth row of the T column.

The first PVT point must be within the range XM[1]…XM[2]. the remaining PVT points need not be within modulo range; but the difference between consecutive PVT position points must be less than(XM[2] - XM[1])/2. For example, suppose that XM[1] = 0 and XM[2] = 1000. If the PVT describes a trajectory beginning at 0 and ending at 10,000, the motor will travel 10,000 counts, fully completing its position range 10 times.

6.1.6.2 Motion Management

In PVT mode, the drive manages a "read pointer" for the PVT table. When the read pointer is N, the present motion segment starts at the coordinates written on the Nth row of the table, and ends at the coordinates of the (N+1) row[8].

When the time period specified by QT[N] elapses, the N segment is complete, the drive increments the read pointer to N+1, and then reads the N+2 PVT table row in order to calculate the parameters of the next motion segment.

The entire PVT table need not be used for a given motion. Its use is defined by the following parameters:

Parameter	Use
MP[1]	Lowest valid row of the PVT table
MP[2]	Highest valid row of the PVT table
MP[3]	A bit field: Bit 0 is: 0: Stop motion if read pointer reaches MP[2] 1: Continue motion when read pointer reaches MP[2]. The next row of the table is MP[1]. Bit 1 is: 0: PVT motion not expected to terminate. Issue an exception if it does. 1: PVT motion expected to terminate. When all data in PVT table has been used, exit PVT mode to Idle mode without issuing an emergency object.

Table 6-8: **PVT Motion Parameters**

The following flowchart depicts the basic PVT mode. It assumes that the commands `PV=N; BG;` were entered, with $1 \leq N \leq 64$.

The command `PV=N` sets the read pointer of the table to `N` and specifies that the next `BG` will start a PVT motion. `BG` then starts the motion.

The PVT table may be written online while PVT motion is being carried out. An infinite non-periodic motion can be generated in cyclical mode (MP[3]=1) by programming the PVT table on-the-fly.

[8] The PVT mode may be cyclical, according to MP[3]. In this case, N+1 must be interpreted in the modulo sense.

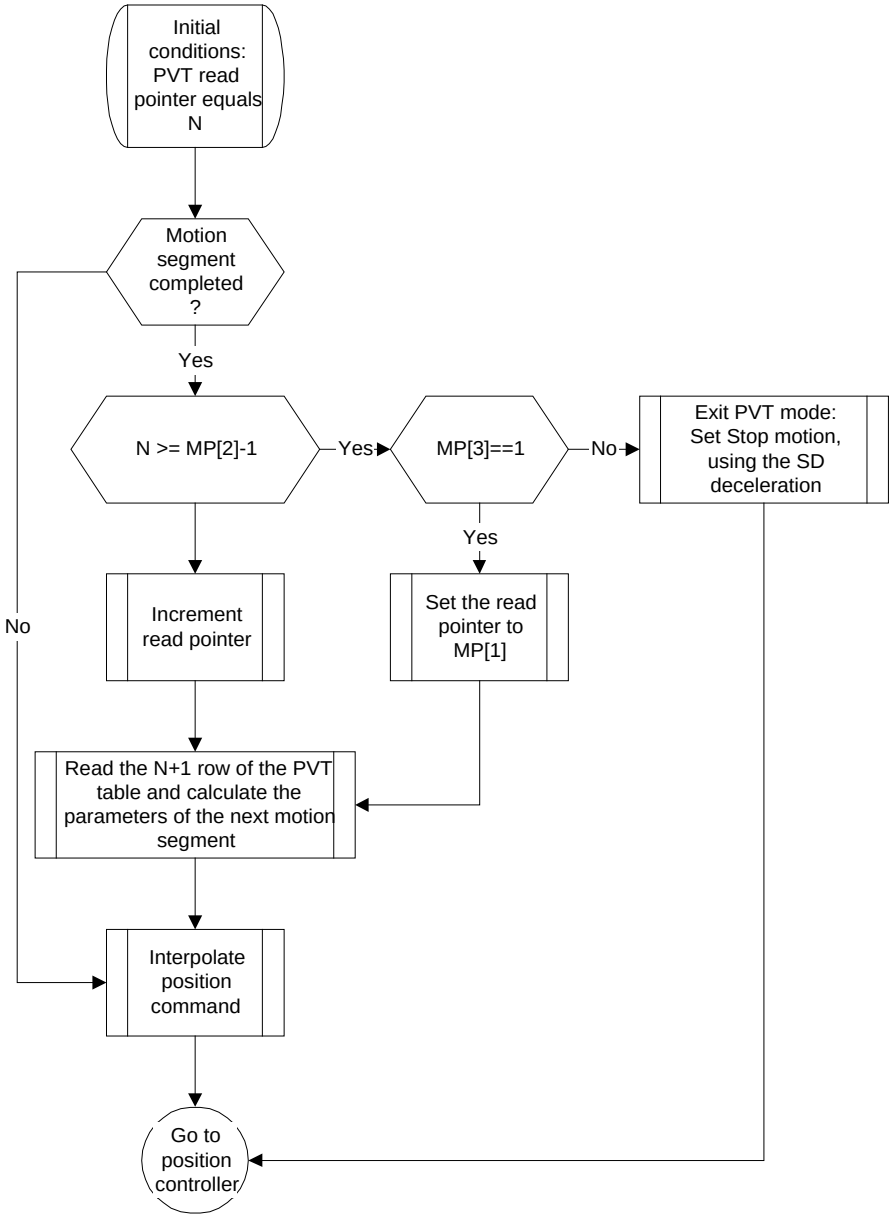


Figure 6-6: **PVT Decision Flowchart**

The host must know how much free space is available in the PVT table in order to continue programming an executing PVT motion. This is achieved most efficiently by tracking the table's read and write pointers. The host is aware of the write table status, because it controls writing to the table. If there is a doubt, the host can query MP[6]. The PV command is used to query the read pointer.

The read and write pointers can be mapped to a synchronous PDO, so that a CAN master can efficiently and continuously stay informed about the status of multiple drives running PVT in parallel in a network. Rather than polling the status of the PVT motion continuously, the host can use the queue underflow CAN emergency object as a request to refill the PVT table.

An unused part of the PVT table may be programmed for the next motion while the present motion is executing. An attempt to modify the data of an executing motion segment generates an error.

6.1.6.3 Mode Termination

PVT motion terminates upon one of the following events:

- The motor is shut down, either by programming MO=0 or by an exception.
- Another mode of motion is set active; by programming PA=xxx; BG, for example. In this case, the new motion command executes immediately, without having to explicitly terminate the PVT mode.
- The PVT motion manager runs out of data. This occurs when the read pointer reaches MP[2] and MP[3] is zero. This may also occur in auto-increment mode (CAN communications only) if the read pointer reaches the write pointer. In that case, the PVT motion is stopped immediately, using the SD deceleration. Note that if the last programmed PVT speed is zero, the PVT motion terminates neatly and the stop at the end of the motion does nothing.

6.1.6.4 PVT Motion Using CAN

The PVT table allows for the performance of both pre-designed and online motion plans. For online motion design, new entries can be written to the PVT table while the PVT motion is executing. The online motion design ability is limited by the speed of the communication interface.

With an RS-232 ASCII communication interface, a single PVT table row is programmed with the following format:
QP[xx] = xxx,xxx,xxx; QV[xx] = x,xxx,xxx; QT[xx] = xxx;
Up to 40 characters may be required to program a single PVT table row. At a communication rate of 19,200 baud, this may take 20 milliseconds. At a communication rate of 115,200 baud, this may take 4 milliseconds.

The CAN communication option allows much faster PVT table programming, by packing an entire PVT table row into one PDO communication packet. For easy synchronization with the host, the drive may be programmed to send the PVT read and write pointers continuously to the host as a synchronous PDO, or to send an emergency object whenever the number of as yet unexecuted motion segments falls below a given threshold.

The PVT Motion Programming Message

An entire row of the PVT table may be programmed by a single PDO — 0x200+ID — where ID is the node ID of the drive. Note that before using this PDO, it must be mapped to the object 0x2001. The PDO is mapped for the PVT mode as follows:

Object dictionary index:	0x2001
Type:	RECORD, three elements
Access:	Write only
Structure:	Signed32 position; Signed24 speed; Unsigned8 time
PDO mapping:	Yes
Value limits	No
Default value:	Not applicable

The PDO does not specify the PVT table row to be programmed; instead, a write pointer specifies the row. The parameter MP[6] initially sets the write pointer. A new PVT CANopen message (object 0x2001) writes the data to the table row indicated by MP[6] and then automatically increments MP[6]. The CANopen auto-increment mode is described in the following flowchart:

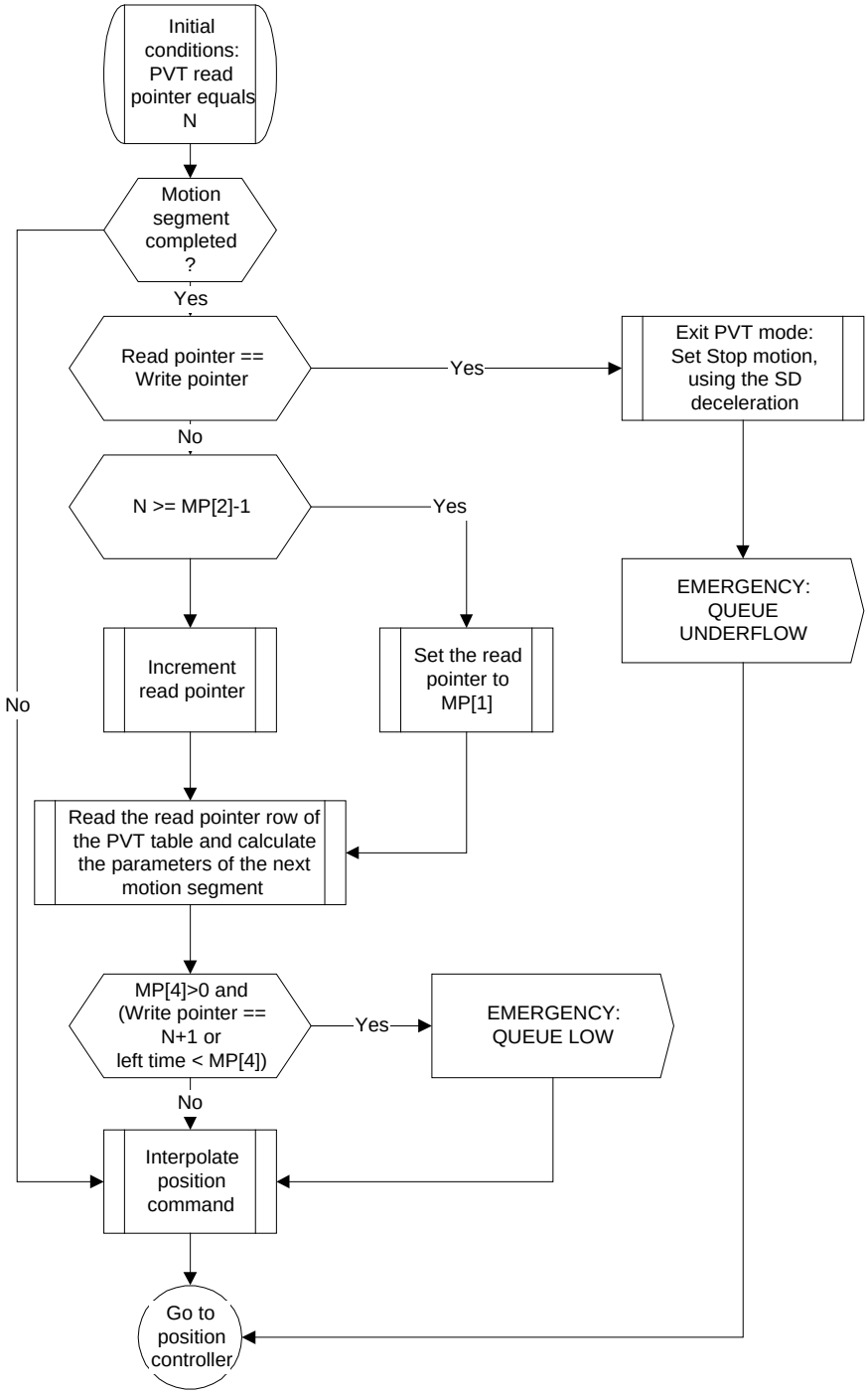


Figure 6-7: **PVT Auto-increment Mode Flowchart**

This flow differs from the basic mode because:

- The motion queue underflow is diagnosed by the read pointer reaching the write pointer.
- Emergency objects are issued for the queue low and queue underflow events.

Programming Sequence for Auto-increment PVT Mode

PVT motion must begin with the initial programming of the PVT arrays. To do so, set:
MP[1] = First valid line in PVT table
MP[2] = Last valid array in PVT table
MP[3] = 1 for cyclical mode
MP[4] = Not relevant for PVT (PT only)
MP[5] = Number of remaining programmed motion rows for issuing a "PVT queue low" emergency object. Set to zero if no "PVT queue low" warning is desired.
MP[6] = Write pointer. This is the next position in the PVT table to be written by the CANopen object 0x2001.

Set the QP[N], QV[N] and QT[N] array elements for at least the first two rows of the PVT table. This is needed because PVT requires at least two time points for interpolating the motion trajectory.

If not already set, set UM=4 or UM=5, and MO=1.
Set PV=N, assuming QP[N], QV[N], QT[N] and QP[N+1], QV[N+1], QT[N+1] are all programmed to valid values.
Start the motion with a BG.
Use the CAN PVT auto-increment command for the rest of the PVT motion.

When object 0x2001 is used to feed PVT reference points to the drive while a PVT motion is executing, the drive automatically enters Online Feed mode, in which the drive is aware of the position written and takes the following precautions:

- If the feed is too slow, so that the drive must fetch unprogrammed points, the motion will abort with a "Queue underflow" emergency.
- If the feed is too fast, so that points not yet used by the drive are overstruck, the drive will reject the feed attempt with a "Queue overflow" emergency.

The Online Feed mode will continue until the end of the current PVT motion and will not be applied to the next motion.

To stay informed about how the PVT motion is advancing, the read and write PVT table pointers can be received continuously, mapped to a synchronous PDO, or the "Queue low" emergency signal can be used to indicate the need for more data. The "Queue low" emergency message includes the present location of the read pointer and the write pointer. It is safe to send more PVT data PDOs until the write pointer is one location before the read pointer specified by the "Queue low" emergency message.

The host is well aware of the location of the write pointer, because it can count its own messages. However, a data message may be rejected because the queue is full, or because a message has been lost. In such a case, the drive issues an emergency object to the host. After receiving the object, the true location of the write pointer may be unclear. The host may then set MP[6] to the possibly-rejected table row and continue writing from there.

MP[5] (the number of rows remaining for "Queue low" emergency) should not be set to too high a value. For example, consider a slow-responding host that manages a 64-row PVT queue in the drive. Suppose that the PVT row times are 10 milliseconds each and that MP[5] = 55. The host receives a "Queue low" emergency object notifying it that there are 64 - 55+1 = 8 free programmable rows in the PVT table. Suppose that the host takes 50 milliseconds to respond, and 5 milliseconds to program each row. The 8 rows will have been programmed after 90 milliseconds. In the meantime, 9 additional PVT table rows will have been executed, and there are only 54 valid rows in the PVT table. Because 54 is lower than MP[5], there will be no more "Queue low" emergency messages until the PVT table is exhausted, and PVT mode is terminated. The situation can be remedied if the host requests the drive for a PV (location of read pointer) after the PVT table writes are complete. If PV < MP[5], the programming process has taken too much time, and the writing must be continued.

Accurate timing with respect to the host is the essence of multiple-axis synchronized motion. Such timing can be achieved by using the CAN SYNC signal and the CAN synchronized BG service, as described in the Elmo *CAN Implementation Manual.*

6.1.6.5 PVT Motion Mode Parameters

The following parameters apply to PVT motion:

Parameter	Use	Comment
UM (Unit Mode)	Units modes 3, 4 and 5 select the position modes.	
SD (Stop Deceleration)	Rate of deceleration when motion is killed by queue underflow or exception. The rate of acceleration to catch up if PVT is started with improper initial connections.	
PV (Position/Velocity/ Time)	Set a PVT motion command	
QP[N], QV[N], QT[N]: PVT table entries	Set values to PVT table	PVT table elements can also be set using PDOs.
MP (Motion Parameters)	MP[1] = First valid row in PVT table. MP[2] = Last valid row in PVT table. MP[3]: Bit0 = Cyclical motion (0: Non-cyclical, 1: Cyclical) Bit1 = Expected stop (0: Issue emergency on stop; 1; Expect stop) MP[5] = Number of yet unexecuted table rows for "Queue low" alarm. MP[6] = Initial value for write pointer	Configure a PT or PVT motion. MP[6] and MP[5] are for CANopen auto-increment mode only.

Table 6-9: **PVT-related Parameters**

The following CAN emergencies are supported:

Error Code (Hex)	Error Code (Dec)	Reason	Data Field
0x56	86	Queue is low. Number of as yet unexecuted PVT table rows has dropped below the value stated in MP[4].	Field 1: Write pointer Field 2: Read pointer
0x5b	91	Write pointer is out of physical range ([1…64]) of PVT table. Reason may be an improper setting of MP[6].	Value of MP[6]
0x5c	92	PDO 0x3xx is not mapped.	
0x34	52	An attempt has been made to program more PVT points than are available in queue.	Field 1: Index of PVT table entry that could not be programmed
0x7	7	Cannot initialize motion due to bad setup data. The write pointer is outside the range specified by the start and end pointers.	
0x8	8	Mode terminated and motor has been automatically stopped (in MO=1).	Data field 1: Write pointer Data field 2: 1: End of trajectory in non-cyclical mode 2: A zero or negative time specified for a motion interval 3: Read pointer reached write pointer
0x9	9	A CAN message has been lost.	

Table 6-10: **PVT CAN Emergency Messages**

6.1.7 Position - Time (PT)

In a PT motion, the user specifies a sequence of absolute positions with equal time spaces to be visited by the drive. The time space must be an integer multiple of the drive sampling time. Between the user-specified positions, the drive interpolates smooth motion. The position specifications are absolute.

6.1.7.1 Interpolation Mathematics

PT implements a third-order interpolation between the position data points provided by the user.

Let $T = m * T_s$

where:
T_s is the sampling time of the position controller. The parameter WS[28] reads T_s.
T is the sampling time of the PT trajectory.
m (system parameter MP[4]) is the integer parameter that relates T_s and T.

For m = 1, no interpolation is required. For m > 1, there are certain sampling instances of the position controller for which the path command must be interpolated, using a third-order polynomial interpolation.

The user provides position points P(k), k = 1…N.
The drive calculates the speeds V(k), k=1…N for points P(k), k=1…N as follows:

- If k is an ordinary point inside the path:
$$V(k) = \frac{P(k+1) - P(k-1)}{2T}$$
- If k is the first programmed point in the path:
$$V(k) = \frac{P(k+1) - P(k)}{T}$$
- If k is the last programmed point in the path:
$$V(k) = \frac{P(k) - P(k-1)}{T}$$

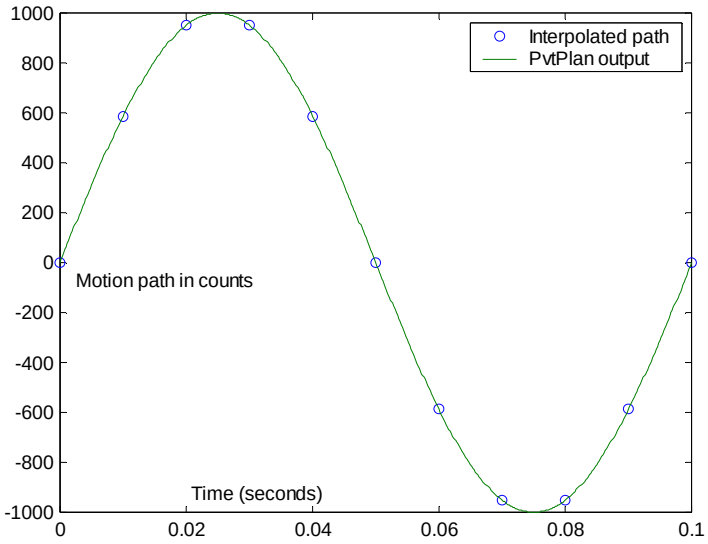
For each motion interval, four requirements must be satisfied:

- Start position
- End position
- Start speed
- End speed

These four requirements exactly suffice for solving the third-order interpolating polynomial.

Example:
Consider the position controller reference signal P(t) - $\sin(2\pi * 10t)$
where t is the time in seconds.

The controller sampling time is 200 microseconds and the path is programmed with a data point once per MP[4] = 50 controller sampling times (10 milliseconds). The PT reference points are depicted as circles in the following graph. The path interpolated by the drive is shown as a solid line.



6.1.7.2 The PT Table

The vector QP[N] defines the position points for PT motion. Each element of the vector defines the position at a given time. The QP vector has 1024 elements, and can therefore specify up to 1023 consecutive PT motion segments, or 1024 PT motion segments in cyclical mode.

The first PT point must be within the range XM[1]…XM[2]. The remaining PT points need not be within modulo range; but the difference between consecutive PT position points must be less than (XM[2] - XM[1])/2. For example, suppose that XM[1] = 0 and XM[2] = 1000. If the PT describes a trajectory beginning at 0 and ending at 10,000, the motor will travel 10,000 counts, completing its position range 10 times.

6.1.7.3 Motion Management

In PT mode, the drive manages a "read pointer" for the QP[N] vector. When the read pointer is N, the present motion segment starts at position QP[N] and ends at QP(N+1)[9]. After MP[4] control sampling times, the drive increments the read pointer to N+1, and reads QP[N+2] to calculate the parameters of the next motion segment.

The entire PT table need not be used for a given motion.

The parameters of a PT motion are summarized in the following table:

Parameter	Use	Comment
MP[1]	Lowest valid element of QP vector.	
MP[2]	Highest valid row of QP vector.	

[9] The PT mode may be cyclical, according to [MP3]. In this case, N+1 must be interpreted in the modulo sense.

Parameter	Use	Comment
MP[3]	0: Motion stops if read pointer reaches MP[2]. 1: Motion continues when read pointer reaches MP[2]. The next row of the table is MP[1].	Cyclical behavior definition.
MP[4]	Number of controller sampling times in each PT motion segment.	

Table 6-11: **PT Motion Parameters**

The following flowchart depicts the basic PT mode:

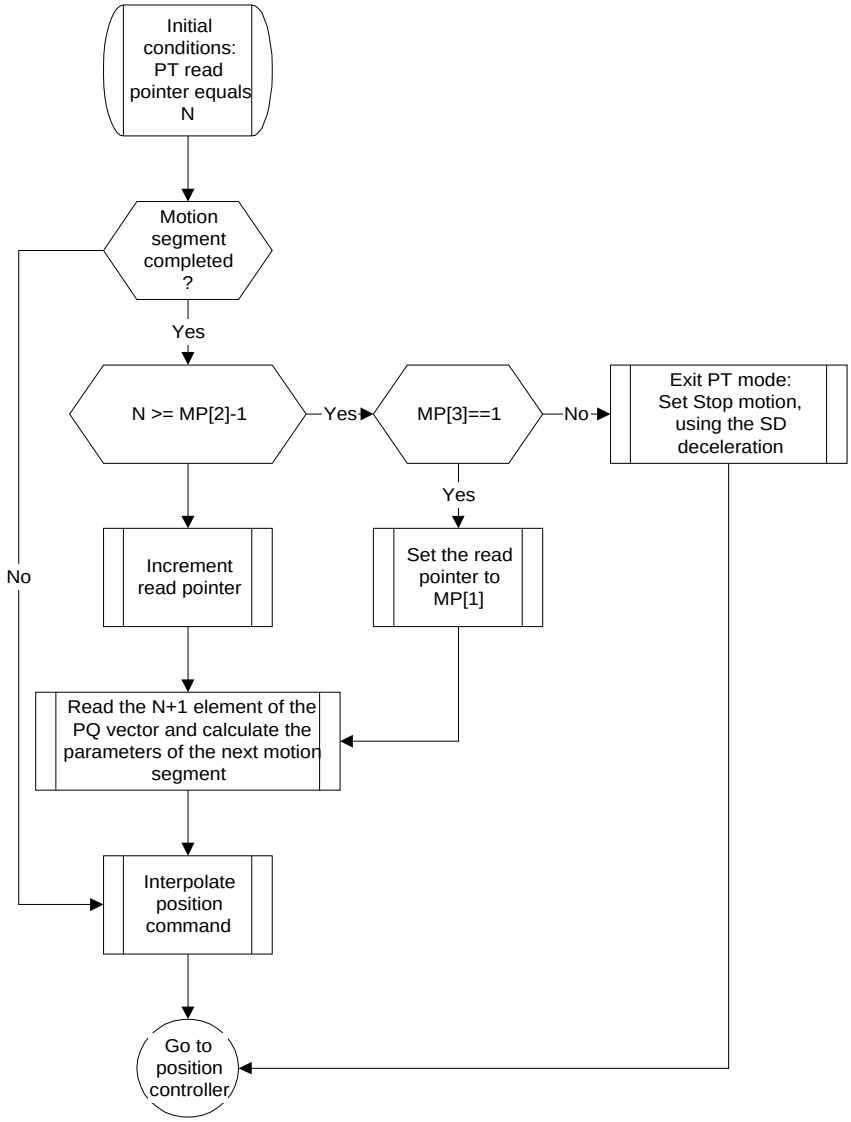


Figure 6-8: **PT Decisions Flowchart**

A PT motion is initiated by stating:
`PT=N` with $1 \leq N \leq 1024$, and `BG`.
The `PT=N` command sets the read pointer of the QP vector to `N`.
`BG` starts the motion.

The QP vector may be written online during a PT motion, as long as no presently-executing PT elements are programmed.

a. Mode Termination

The PT motion terminates when one of the following occurs:

- The motor is shut down, either by programming MO=0 or by an exception.
- Another mode of motion is set active; by programming PA=xxx; BG, for example. In this case, the new motion command executes immediately, without having to explicitly terminate the PT mode.
- The PT motion manager runs out of data. This occurs when the read pointer reaches MP[2] and MP[3] is zero. This may also occur in auto-increment mode (CAN communications only) if the read pointer reaches the write pointer. In that case, the PT motion is stopped immediately, using the SD deceleration. Note that if the last programmed PT speed is zero, the PT motion terminates neatly.

b. PVT Motion Using CAN

The PT table allows for the performance of both pre-designed and online motion plans by writing the QP vector while PT motion is executing. The online motion design ability is limited by the speed of the communication interface. With an RS-232 ASCII communication interface, a single QP vector is programmed with the following format:
QP[xx] = xxx,xxx,xxx;
Up to 18 characters may be required to program a single position point. At a communication rate of 19,200 baud, this may take 9 milliseconds. At a communication rate of 115,200 baud, this may take 1.8 milliseconds.

The CAN communication option allows must faster PT programming, by packing two position points into one PDO communication packet. For easy synchronization with the host, the drive may be programmed to send the PT read and write pointers continuously to the host as a synchronous PDO, or to send an emergency object whenever the number of yet unexecuted motion segments falls below a given threshold.

c. The PT Motion Programming Message

Two positions for the QP vector can be programmed in the eight bytes of a single PDO — 0x300+ID — where ID is the node ID of the drive. Note that before using this PDO, it must be mapped as follows:

Object dictionary index:	0x2002
Type:	RECORD, two elements
Access:	Write only
Structure:	Signed32 Position1; Signed32 Position2
PDO mapping:	Yes
Value limits	No
Default value:	Not applicable

The PDO does not specify the QP vector elements to be programmed; instead, a write pointer specifies them. The parameter MP[6] sets the value of the write pointer, which may be set once for the entire motion. The write pointer is incremented automatically by 2 each time the drive receives a new PT motion-programming message. The CANopen auto-increment mode is described in the following flowchart:

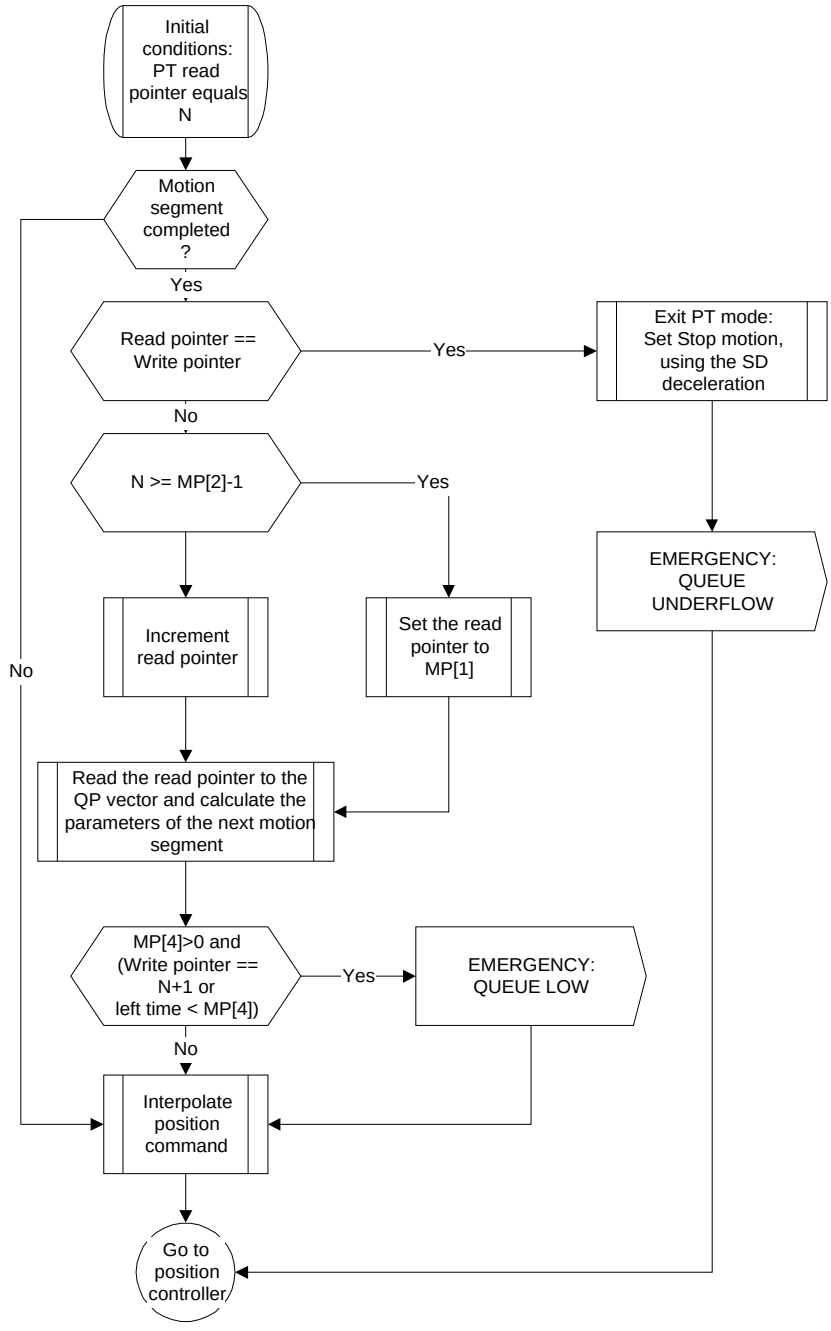


Figure 6-9: **PT Auto-increment Mode Flowchart**

This flow differs from the basic mode because:

- The read pointer reaching the write pointer identifies motion queue underflow.
- Emergency objects are issued for the queue low and queue underflow events.

d. Programming Sequence for Auto-increment PVT Mode

PT motion must begin with the initial programming of the PT arrays. To do so, set:
MP[1] = First valid line in PT table.
MP[2] = Last valid array in PT table.
MP[3] = 1 for cyclical mode.
MP[4] = The ratio between the length of the PT time interval and the sampling time of the position controller.
MP[5] = Number of as yet unused QP[N] elements when a "PT queue low" emergency object is sent. Set to zero if no "PT queue low" warning is desired.
MP[6] = Write pointer. This is the next position in the QP[N] vector to be written by the CANopen object 0x2002.

Set QP[N] for at least the first two points in the PT motion. This is needed because the PT algorithm requires at least two time points for interpolating the motion trajectory. Set at least three points if the speed at the second point is to be continuous.

If not already set, set UM=4 or UM=5, and MO=1.
Set PT=N, where QP[N] and QP[N+1] — and preferably QP[N+2] — are all programmed to valid values.
Use the CAN PT auto-increment command for the rest of the PT motion.

When object 0x2002 is used to feed PT reference points to the drive while a PT motion is executing, the drive automatically enters Online Feed mode, in which the drive is aware of the position written and takes the following precautions:

- If the feed is too slow, so that the drive must fetch unprogrammed points, the motion will abort with a "Queue underflow" emergency.
- If the feed is too fast, so that points not yet used by the drive are overstruck, the drive will reject the feed attempt with a "Queue overflow" emergency.

The Online Feed mode will continue until the end of the current PT motion and will not be applied to the next motion.

To stay informed about how the PT motion is advancing, the read and write PT table pointers can be received continuously, mapped to a synchronous PDO, or the "Queue low" emergency signal can be used to indicate the need for more data. The "Queue low" emergency message includes the present location of the read pointer and the write pointer. It is safe to send more PT data PDOs until the write pointer is one location before the read pointer specified by the "Queue low" emergency message.

The host is well aware of the location of the write pointer, because it can count its own messages. However, a data message may be rejected because the queue is full, or because a message has been lost. In such a case, the drive issues an emergency object to the host. After receiving the object, the true location of the write pointer may be unclear. The host may then set MP[6] to the possibly-rejected table row and continue writing from there.

6.1.7.4 PT Motion Mode Parameters

The following parameters apply to PT motion:

Parameter	Use	Comment
UM (Unit Mode)	Units modes 3, 4 and 5 select the position mode.	
SD (Stop Deceleration)	Rate of deceleration when motion is killed by queue underflow or exception. SD is also the acceleration to catch up with a PT motion started with improper initial connections.	
PV (Position/Time)	Set a PT motion command.	Special features are available for PT using CAN communication.
QP[N]: PT table entries	Set values to PT table.	
MP (Motion Parameters)	MP[1] = First valid row in PT table. MP[2] = Last valid row in PT table. MP[3]: Bit0 = Cyclical motion (0: Non-cyclical, 1: Cyclical). MP[4] = Ratio between command sampling time and position controller sampling time. MP[5] = Time for "Queue low" alarm. MP[6] = Initial value for write pointer.	Configure a PT or PVT motion. MP[6] and MP[5] are for CANopen auto-increment mode only.
WS[28]	Sampling time of the position controller, in microseconds.	A read-only parameter. WS[28] is an integer multiple of the basic sampling time as set by TS.

Table 6-12: **PT-related Parameters**

The following CAN emergencies are supported:

Error Code (Hex)	Error Code (Dec)	Reason	Data Field1
0x56	86	The time for the entire remaining valid PT program has dropped below the value stated in MP[4].	Time (milliseconds) remaining with valid motion program.
0x5b	91	Write pointer is out of physical range ([1…1024]) of QP vector. Reason may be an improper setting of MP[6].	Value of MP[6]

Error Code (Hex)	Error Code (Dec)	Reason	Data Field1
0x5c	92	PDO 0x3xx is not mapped.	
0x34	52	An attempt was made to program more PT points than supported by queue.	Index of PT table entry that could not be programmed
0x7	7	Cannot initialize motion due to bad setup data. The write pointer is outside the range specified by the start pointer and the end pointer.	
0x8	8	Mode terminated and motor has been automatically stopped (in MO=1). Reasons: End of trajectory in non-cyclical mode (MP[3] = 0) [additional code 1] A zero or negative time was specified for a motion interval [additional code 2] Read pointer reached write pointer [additional code 3]	Read pointer.

Table 6-13: **PVT CAN Emergency Messages**

6.2. The External Position Reference Generator

This section summarizes how the drive generates the external motion command for the position controller. The external position reference is useful for:

- Positioning a manipulator on a moving object. The desired position of the manipulator with respect to the object on the conveyor can be programmed as a software motion, such as PTP or PVT. The position of the conveyor is not known in advance and must be measured online; using the auxiliary encoder input, for example. The reading of the auxiliary input is scaled by the follower ratio parameter (FR[3]), and added to the software command.
- Driving the drive as a slave in a larger arrangement. For example, when the drive moves a valve in a pressure control system, its position command may be an analog output of a pressure controller.
- Synchronizing several drives, all driven by the same auxiliary encoder signal. Each of the drives uses its ECAM table to derive its own motion path from the auxiliary signal.

The external position reference is generated by the following scheme:

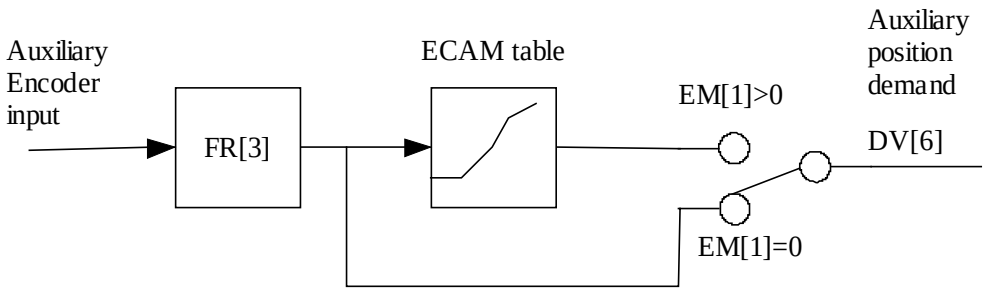


Figure 6-10: **External Position Reference Generator**

The following parameters determine the composition of the position reference:

Parameter	Action
FR[3]	Scale the auxiliary encoder input. Applicable only if the auxiliary encoder is not used for position feedback.
EM[1]	Define whether the ECAM table transforms the external reference: EM[1]=0: Do not use ECAM table. EM[1]=1: Use linear ECAM table to transform external command. EM[1]=2: Use cyclical ECAM table to transform external command.
RM	Define whether an external reference is used: RM=0: Do not use external reference. RM=1: Use external reference.
DV[6]	Reads the external position reference.

Table 6-14: **Position Reference Parameters**

6.2.1 Follower

In Follower mode (RM=1, EM[1]=0), the external speed command tracks the auxiliary encoder speed at a ratio of FR[3], as depicted in Figure 6-11. In this figure, the auxiliary encoder counts the PY modulo in the range [0…500]. The derived external position reference advances at the same rate as the auxiliary encoder while FR[3]=1. Jumps in PY due to the modulo count are not reflected in auxiliary position reference DV[6]. When FR[3] changes to 2, DV[6] advances at twice the speed of PY.

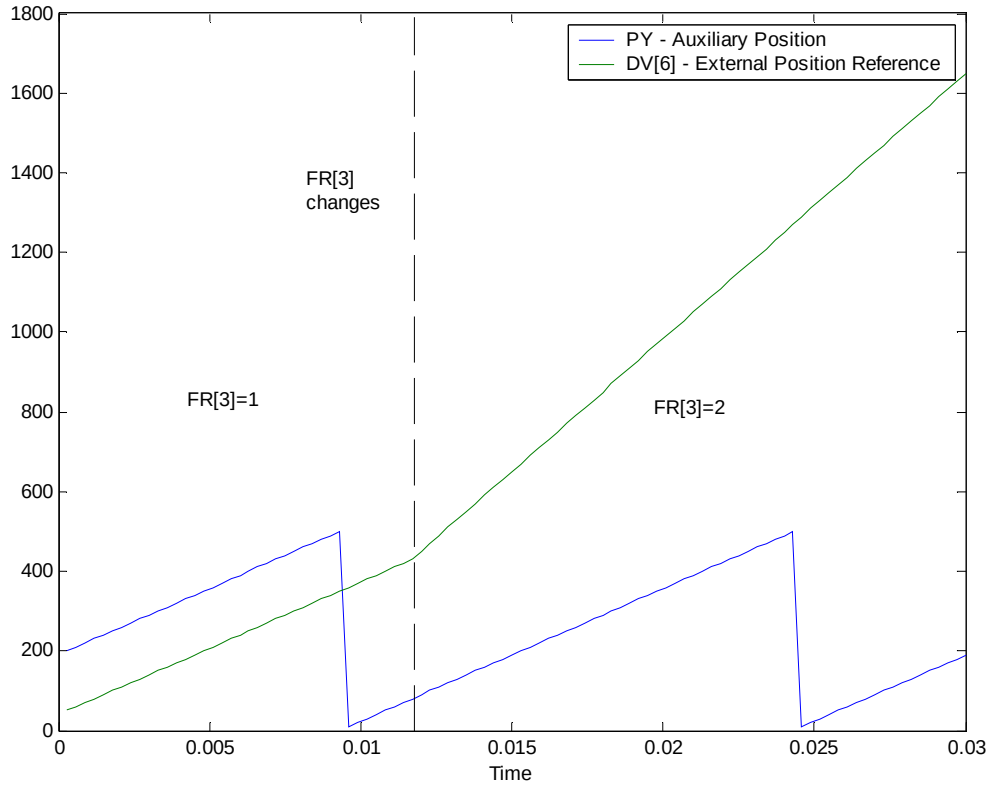
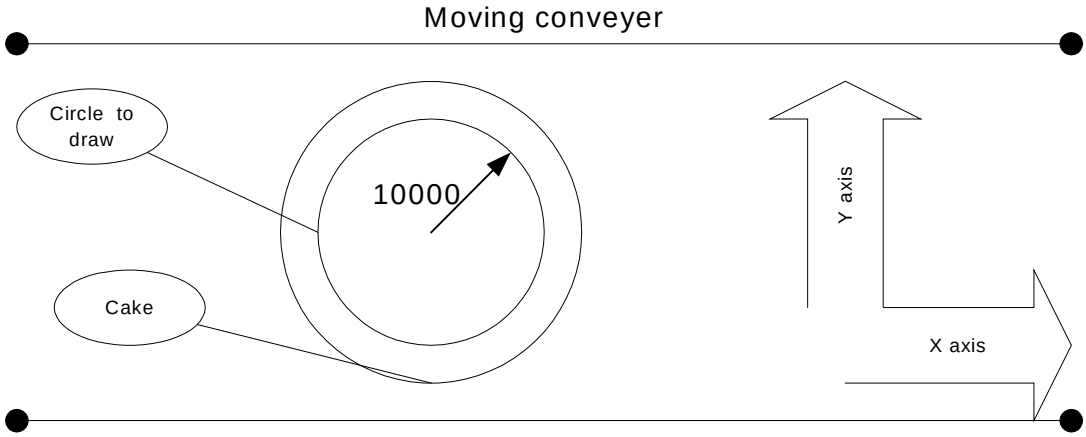


Figure 6-11: **Follower Ratio**

Example:
This example illustrates how a moving object is handled by the application:

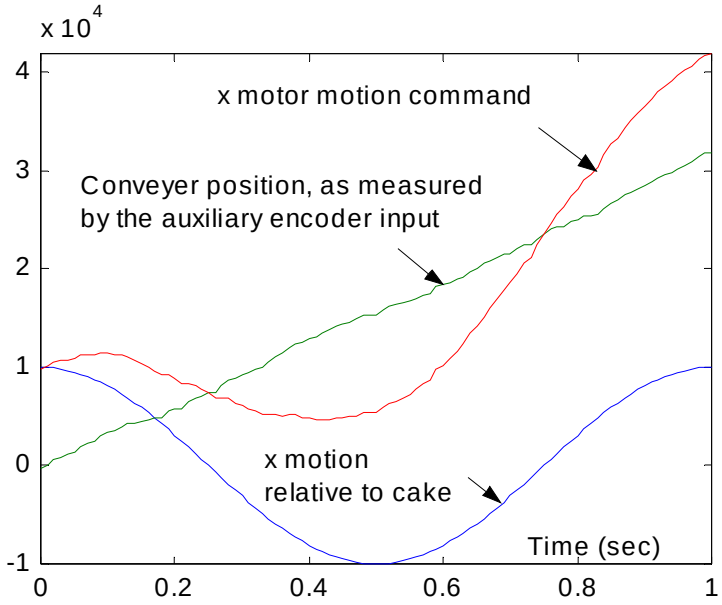


In this application, an x-y stage draws a chocolate picture on a cake while the cake travels on a conveyor. The drawing must be accurate with respect to the cake.

In order to draw a circle of radius 10,000 encoder counts on the cake in one second, the x-axis motor must follow the trajectory:
$x(t) = 10{,}000 * \cos(2\pi t) + c(t)$
where c(t) is the position of the conveyor.

If the conveyor has an encoder, it can be used to compensate for its motion.

Suppose that the resolution of the conveyor encoder is similar to the resolution of the x-axis encoder. To draw an exact circle on the moving cake, the motion 10,000 * cos(2πt) is programmed as PVT and RM=1, FR[3]=1.



6.2.2 ECAM

ECAM is an acronym for electronic CAM, in which the position reference to the drive is not directly proportional to the total external inputs, but is rather a function of them.

The ECAM-related commands are as follows:

Commands	Action
EM[1]	Indicates whether the ECAM function is active: 2: ECAM is cyclical. 1: ECAM is linear. 0: Direct external referencing. Set EM[1] to synchronously activate the most recent settings of EM[2], EM[3], EM[4], EM[5] and EM[7].
EM[2]	Last valid index of ECAM table. Maximum for EM[2] is 1024. The EM[2] setting goes into effect only at next setting of EM[1] or next MO=1.
EM[3]	Starting position: value of the input to the ECAM function for which the output of the ECAM function will be ET[EM[5]] (ET of EM[5]). If EM[3] is out of range for PY, it will be taken modulo PY.
EM[4]	Table difference (see sections 6.2.2.1 and 6.2.2.2). The EM[2] setting goes into effect at next setting of EM[4] or next MO=1.

Commands	Action
EM[5]	First valid index of ECAM table. The EM[2] setting goes into effect at next setting of EM[5] or next MO=1.
EM[6]	Write pointer for fast loading of the ECAM table via CAN bus (see section 6.2.4).
EM[7]	Last interval shortening. EM[7] allows the ECAM table length (ECAM cycle in the EM[1]=2 cyclical mode) not to be an integer multiple of EM[4] (see sections 6.2.2.1 and 6.2.2.2).
EM[8]	Reports present executing segment of ECAM table. Used to determine if an action is executing or is complete.

Table 6-15: **ECAM-related Commands**

The input to the ECAM table (IET) is FR[3] * (auxiliary encoder - EM[3]).

The ECAM table defines the external position reference for IET values between 0 and IETmax = (EM[2] - EM[5]) * EM[4] - EM[7].

6.2.2.1 Linear ECAM

Linear ECAM is selected by EM[1]=1. In this mode:

- If IET < 0, the external position command (output of the ECAM table) will be ET[EM[5]].
- If IET > IETmax, the external position command will be ET[EM[2]].
- If 0 < IET < IETmax, the external position command will be derived by linear interpolation of the ECAM table.

The output of the ECAM table does not need to be in the range of the main position sensor (refer to the XM[N] command section in the *SimpIIQ for Steppers Command Reference Manual*). If the output of the ECAM table is out of the position sensor range, it is interpreted modulo the position sensor range.

Example 1:
The following figure illustrates the behavior of linear ECAM for EM[5]=1 and EM[2]=4.

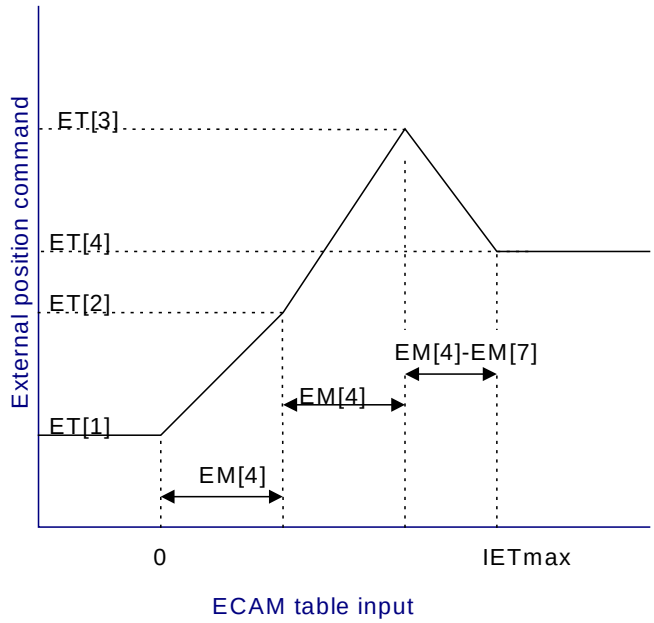
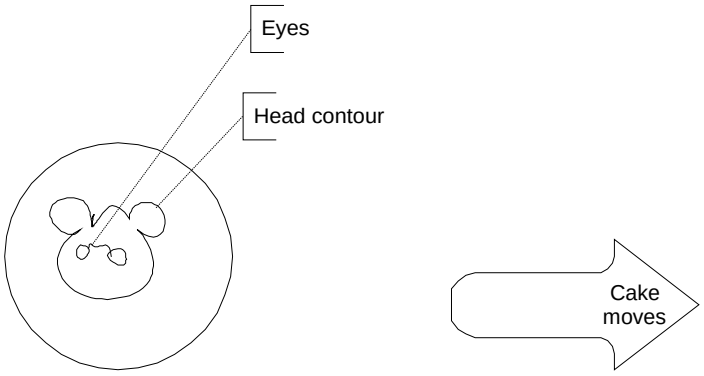


Figure 6-12: **Linear ECAM**

Example 2:
Consider an application in which a two-axis x-y servo system is used to plot chocolate bears on birthday cakes:



The cakes come from the oven on a conveyor. An incremental encoder measures the position of the conveyor. Each time a new cake arrives, it activates the DIN#1 input of the drive. In response to DIN#1, the x-y axes begin to contour the head of the bear, drawing it with chocolate. The chocolate flow is stopped while the x-y axis travels towards the start of the bear eyes. After the eyes are drawn, the x-y stage returns to its initial position, ready for another cake.

Both drives that manage the x and y axes operate in ECAM mode. They get their position reference as a function of the location of the conveyor, via the auxiliary encoder input. The ECAM motion starts when a cake arrives at the plotting station and continues until the conveyor travels 4000 counts.

One of the drives uses a digital output to control the flow of the chocolate out of the drawing nozzle.

The drive program is:

`IL[1]=7`	Program DIN#1 as general-purpose input.
`EM[1]=1`	Enable ECAM.
`EM[2]=200`	Length of ECAM vector.
`EM[3]=0`	Starting position.
`EM[4]=100`	Conveyor encoder counts between two consecutive ECAM table entries.
`ET[1]=…;ET[2]=…;ET[100]…;`	Program numeric data of ECAM table. If ECAM table data is changed from cake to cake, consider loading ECAM table using fast CAN loading method.
`UM=5`	Set single sensor position mode.
`RM=1`	Enable external referencing.
`FR[3]=0;`	Kill external input.
`MO=1`	Start motor.
`PA=1000;BG`	Go to waiting position.
`HY[2]=0;HY[3]=9;HY[1]=1`	Null auxiliary encoder count upon cake arrival (DIN#1 high).

Each drive has the following AUTO_I1 routine:

`function AUTO_I1`	DIN#1 has operated an already-programmed auxiliary homing process to synchronize the following input.
`OB[1]=1;`	Activate chocolate flow.
`FR[3]=1;`	Enable auxiliary encoder input with follower ratio of 1.
`until (PY >= 2000)`	Wait until end of head contour.
`OB[1]=0;`	Stop chocolate.
`until (PY >= 3000)`	Go to start of eyes.
`OB[1]=1;`	Restart chocolate.
`until (PY >= 4000)`	Draw eyes.
`OB[1]=0`	Stop chocolate.
`FR[3]=0;PA=1000;BG;`	Return to starting point.
`HY[1]=1`	Program auxiliary encoder to reset at next cake.
`return`	End of auto subroutine.

6.2.2.2 Cyclical ECAM

Cyclical ECAM mode is selected by EM[1]=2. In this mode, the master (auxiliary encoder input) can advance indefinitely. The ECAM table defines the slave (motor position) command for one master period, which is the input range to the ECAM table:

IETmax = (EM[2]-EM[5]) × EM[4] – EM[7]

In each master period (in which the master completes a travel of IETmax/FR[3] counts), the slave advances by ET[EM[2]] - ET[EM[5]]. The following figure illustrates the behavior of cyclical ECAM for EM[5]=1 and EM[2]=4.

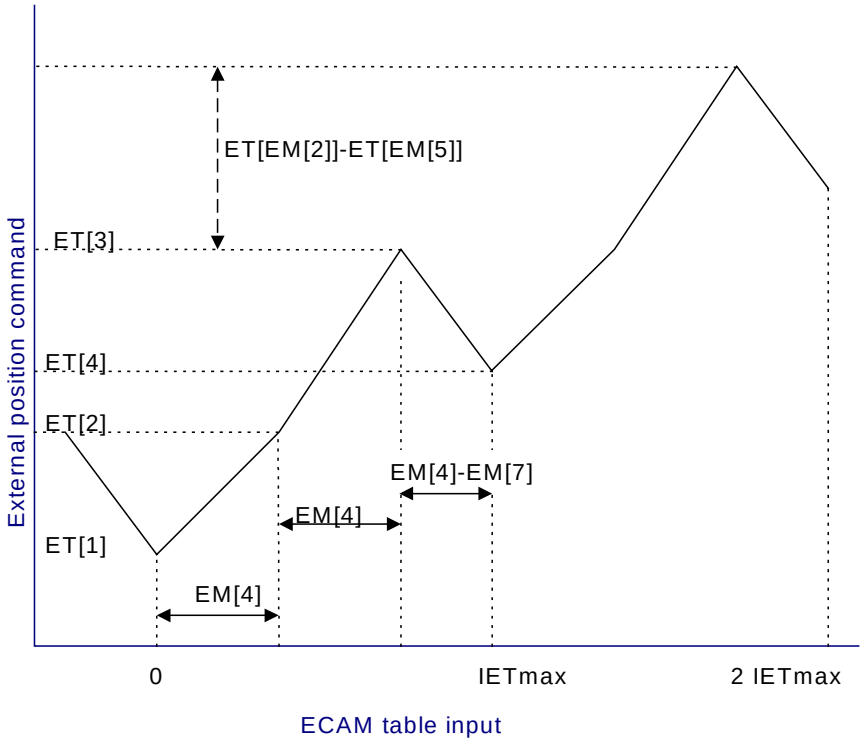


Figure 6-13: **Cyclical ECAM**

Note that the external position command is summed from the ECAM table outcome and a cumulative offset, which is **n** x (*ET*[*EM*[2]] = *ET*[EM[5]]) with **n** being an integer.

The cumulative offset is lost in the following circumstances:

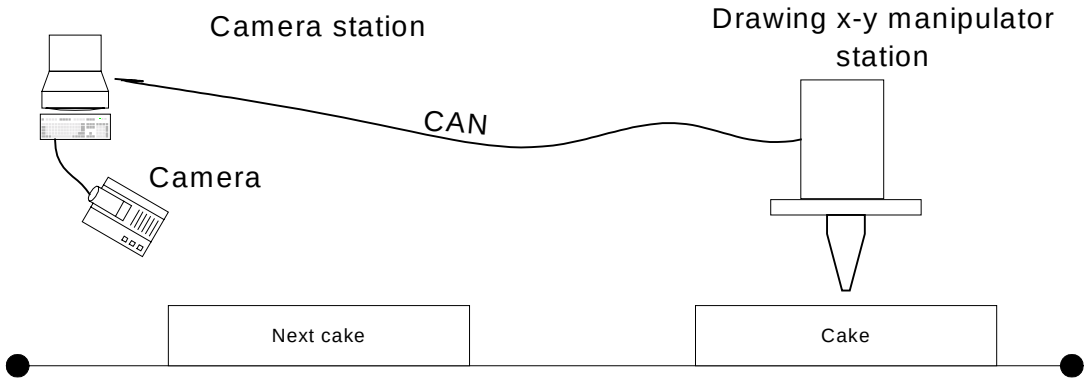
- Homing changes the auxiliary position counter value.
- FR[3] is changed.
- EM[1] is changed.
- EM[1] is set to the existing value, with EM[2], EM[3], EM[4] or EM[5] changed.

When the cumulative offset is lost, and if the software reference generator is not running an interpolated mode, the software reference is automatically adjusted by the same value so that the motor will not jump (refer to section 6.2.5.1).

6.2.3 Dividing the ECAM Table into Logical Portions

The ECAM table can store several distinct movements, with a portion of the table used for each movement. This enables a future movement to be programmed into the drive while the present movement is being executed.

Example:
In the previous example of the chocolate bear, it was assumed that the bear drawing could be programmed one time only. In many food applications, however, the products to be worked on (the cake in the example) are not placed precisely on the conveyor or their shapes may be irregular. A camera images the next coming product and the image is analyzed to form the next motion path. The motion path for the analyzed image is communicated to the drive by the system controller while the drive works on a previously-programmed path, as depicted in the following figure:



Assume that the drive at the x-y manipulator station runs ECAM table entries ET[1]…ET[100]. In the meantime, the system controller programs the next shape to run ET[101]…ET[200]. The preferred method to achieve this is to use the fast CAN ECAM points protocol. While the x-y manipulator works on ECAM table entries ET[101]…ET[2000], the system controller programs ET[1]…ET[100] again, and so on.

The AUTO_RLS program of the previous example is now slightly modified to:

```
function AUTO_I1
```
DIN#1 has operated an already-programmed HY homing process to synchronize the following input.

```
EM[5]=1 + NextSegment * 100; EM[2]=100 * (1 + NextSegment);
EM[1]=1…OB[1]=1; FR=1;
```
The system controller manipulates the user program variable NextSegment to signal which part of the table to use next. The AUTO_I1 line selects the ECAM table portion to use by NextSegment, and then sets EM[1]=1 to activate the new used portion, activating the follower gain and the chocolate flow.

```
until (PY >= 2000)
```
Wait until end of head contour.

```
…
```
Continue as in previous example.

```
return
```

6.2.4 Fast ECAM Programming Using CAN

ECAM table points can be programmed via a fast, auto-increment PDO service. Two positions of the ET table can be programmed in the eight bytes of PDO 0x300+ID, where ID is the node ID of the drive. Note that before using the PDO for fast ECAM table programming, it must be properly mapped[10], as in the following table:

Object dictionary index:	0x2003
Type:	RECORD, two elements
Access:	Write only
Structure:	Bytes [0…3]: Signed32 Position1 Bytes [4…7]: Signed32 Position2
PDO mapping:	Yes
Value limits	No
Default value:	Not applicable

The PDO writes the ET vector at the positions specified by EM[6]. After the PDO, EM[6] increments automatically so that the next PDO will write consecutive places in the ET vector. For example, if EM[6]=10, writing CAN object 0x2003 will fill data into ET[10] and ET[11]. After writing, EM[6] will automatically update to 12.

If ET[EM[6]+1] cannot be written (EM[6]=1024), only position 1 is used to program ET[EM[6]]. Position 2 is ignored and EM[6] increments by 1.

Fast writes to the ECAM table are not protected against writing to the active part of the ECAM table.

Errors in fast ECAM programming issue the following emergency objects:

Error Code (Hex)	Error Code (Decimal)	Reason	Data Field 1
0x5b	91	EM[6] out of physical range ([1…1024]) of ET vector, or an attempt to write into an executing part of an ECAM table.	Value of EM[6]
0x5c	92	PDO 0x3xx is not mapped.	

6.2.5 Initializing External Position Reference Parameters

The external reference generator is initialized at MO=1, and each time a relevant parameter (FR[3], EM[1] or PY) is changed. Note that changing EM[2], EM[3], EM[4], EM[5] and EM[7] has no immediate effect. Setting EM[1] activates the entire set of ECAM parameters.

---

[10] The other mapping options of this PDO write to the PT and PVT motion tables.

6.2.5.1 Jump-free Motor Starting Policy

Upon starting a motor using the MO=1 command, the motor should never jump. The first and most important reason is safety. The other reason is to avoid an excessive position error fault immediately after the motor is started.

To prevent the motor from jumping, the initial software reference is automatically set to the present position of the motor, and the software command remains stationary until a motion instruction is accepted. After setting MO=1, the motion mode is idle so that sending a BG command without the prior specification of another motion mode will not launch any motion.

Example 1:
Suppose that `MO=0, RM=0 and PX=1000` (motor is off, no external reference, present position is 1000 counts). Entering `MO=1` will automatically set the software position reference to 1000 counts, in Idle mode. The command sequence `PA=0;BG` will launch a PTP motion from the position of 1000 counts to the zero position.

Example 2:
Suppose that `MO=0, RM=1, FR[3]=1, PY=3000` and `PX=1000` (motor is off, external reference is generated by auxiliary encoder input with a unit follower ratio, the auxiliary position is 3000 and the present position is 1000 counts). Entering `MO=1` will automatically set the software position reference so that the total position reference will equal PX, Therefore, the initial, automatic software command will be:
`PX - FR[3] * PY = 1000 - 1*3000 = -2000`.

If the auxiliary encoder input is stable, the command sequence `PA=0,BG` will launch a PTP motion from the position of 1000 counts to the position of 3000.

6.2.5.2 On-the-fly Switching of External Reference Parameters

The parameters of the external reference generator can be changed on-the-fly only if the software reference generator is idle. This can be verified by the motion status: MS should return 0 or 1. When the parameters of the external reference generator are changed, the software reference is set so that the motor will not jump at the instant the parameters are set. The setting of the software reference is similar to that described in section 6.2.5.1.

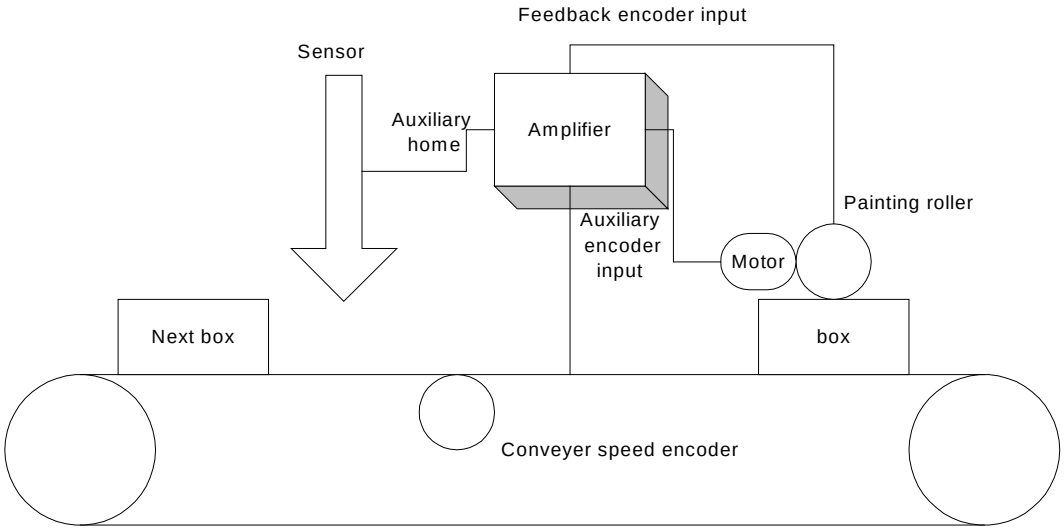
The external reference parameters that can be changed on-the-fly are:

Parameter	Description
FR[N]	Follower ratio
EM[N]	ECAM parameters
HY[N]	Reading of auxiliary encoder

Table 6-16: **External Reference Parameters for On-the-fly Changes**

Example:
Consider a manipulator that works a conveyor. Whenever a box arrives, the roller prints a label on the box. The roller turns continuously at the following line speed:
FR[3] = (counts per roller revolution) / (conveyor encoder counts/mm*2*π*r(mm)).

In order to print the label in the correct place, the roller position must be zero at the point in which the sensor senses the next box. When a new box arrives, it homes the auxiliary encoder to read zero.



At the time of auxiliary homing (when a new box arrives at the sensor), the external reference jumps by PY*FR. The software command to the roller jumps in parallel by -PY*FR, so that the roller continues to turn normally.

Programming the ##AUTO_HM routine as:

```
function AUTO_HM()
PA=0;BG;
return
```

will position the roller phased correctly with respect to the next box.

6.3. Stop Manager

6.3.1 General Description

The stop manager block as the following functions:

- Stops the motion upon sensing a Stop switch, or upon sensing an RLS or FLS limit switch.
- Protects against discontinuity in the controller command. A discontinuity may occur due to:
  - A switch that abruptly stops the motion.
  - An application error (an absolute motion mode such as PVT is started with unacceptable initial conditions).

- Limits the magnitude of the controller command to the maximum allowed range. This is necessary because even if the software command is generated within the permitted limits and the external command is also within the permitted limits, their total value may exceed the permitted limits.

The stop manager prevents the position reference generator from driving the motor to undesired positions. It does not affect the reference generator.

The commands relevant to the stop manager are:

Command	Description
SD	Maximum rate that the motor can accelerate/decelerate
IL[N]	Input logic: define the functions associated with the digital inputs
VH[N], VL[N]	Maximum allowed controller command
XM[N], YM[N]	Modulo count for main and auxiliary sensors

Table 6-17: **Stop Manager Commands**

6.3.2 Stop Manager Internal Elements

The stop manager is depicted in the following diagram:

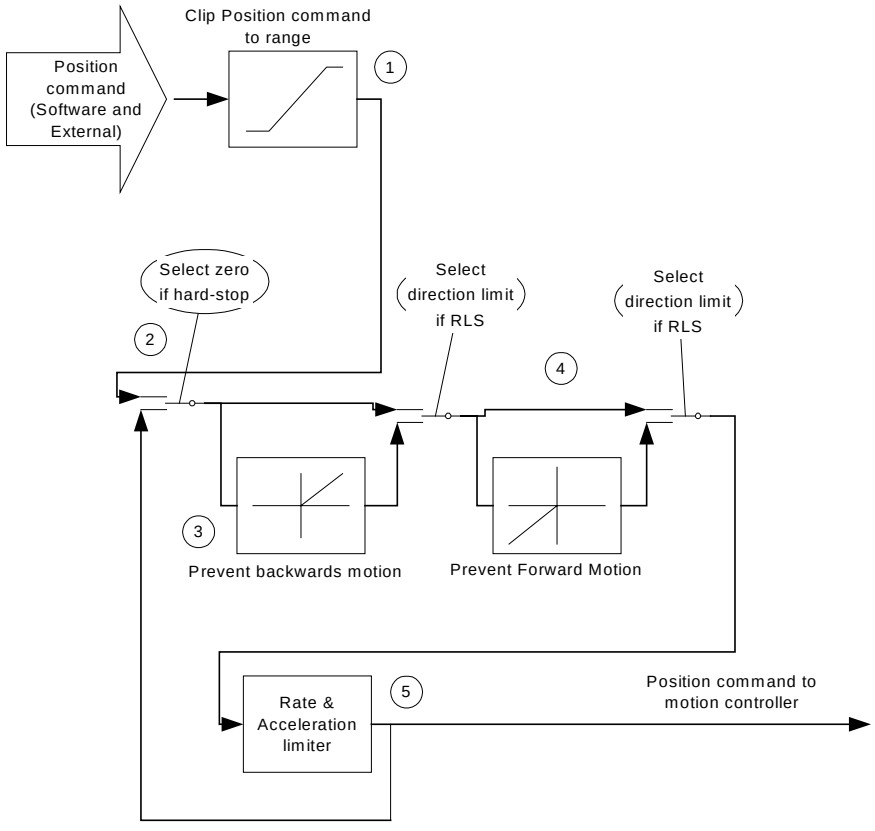


Figure 6-14: **Stop Manager Block Diagram**

Position Command Clipping (No. 1 in Figure 6-14)

The position command is clipped to the following values:

- VH[3] above and VL[3] below, if the position feedback sensor counts linearly
- XM[2] above and XM[1] below in UM=4
- YM[2] above and YM[1] below in UM=5

The clipping is necessary because, in some circumstances (explained previously), the sum of the software command and the external command, or even the software command alone, may exceed the command limits.

Hard Stop (No. 2 in Figure 6-14)

This block stops the desired position reference to its present position if a Hard Stop switch is sensed.

Right Limit Switch (No. 3 in Figure 6-14)

This block stops the desired position reference to its present position if an RLS switch is sensed, and if the output of the position reference generator is less than the controller position command.

Forward Limit Switch (No. 4 in Figure 6-14)

This block stops the desired position reference to its present position if an FLS switch is sensed, and if the output of the position reference generator is greater than the controller position command.

Rate and Acceleration (No. 5 in Figure 6-14)

This block limits the speed and acceleration of the controller position command. The block limits both the acceleration and the deceleration to the SD parameter, and the speed command (the derivative of the position command) is limited to VL[2] from below and VL[3] from above.

The rate and acceleration limiter block intervenes in the following situations:

- The position command to the controller experiences an abrupt change; for example, when a Hard Stop switch is sensed or when a Hard Stop switch is released.
- The reference generator tries to control the motor in the permitted position range, but at too great a speed.
- The position command moves towards its permitted boundary (VL[3] or VH[3]) at a speed greater than can be braked until the boundary with SD acceleration. For example, near VH[3], the upper speed limit may be much lower than VH[2]. When the position command equals VH[3], the maximum allowed speed command is zero.

Example:
The software reference generator generates a sine, using PVT.
The low position limit is `VL[3]= -5000`. The input to the stop manager and its output (the position command to the controller) are depicted in the following figure:

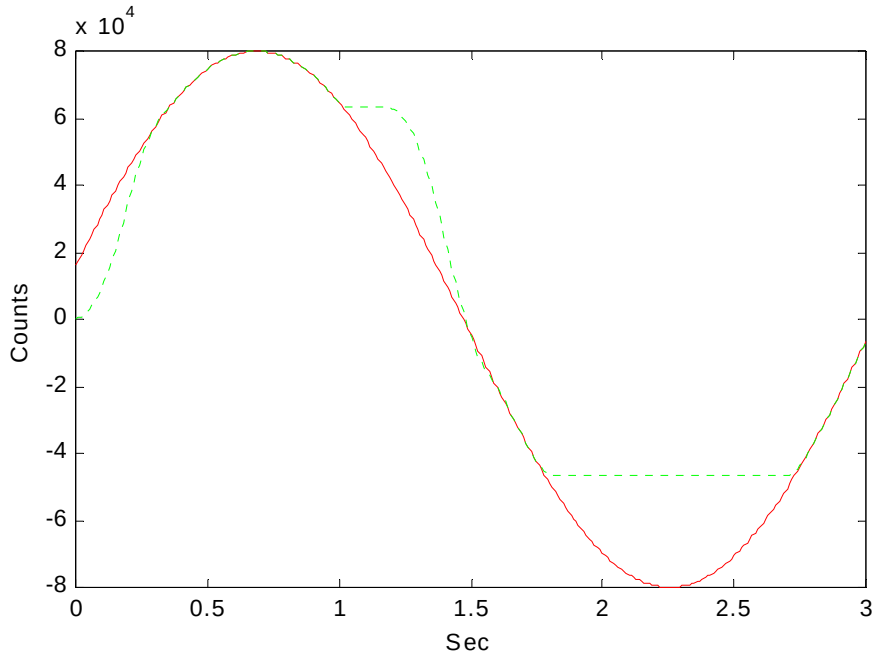


Figure 6-15: **Position Output of the Stop Manager**

The input of the stop manager (the output of the position reference generator) is the red solid line, and the output of the stop manager is the green dashed line. At the time of 0, the PVT starts at position 17,000. The stop manager catches up with the sinusoidal command, with an acceleration limit of SD. At the time of 1 second, a Hard Stop switch becomes active and remains active until the time of 1.2 seconds. The stop manager decelerates the position command to zero speed, using the SD deceleration. At the time of 1.2 seconds, the stop manager begins to catch up again with the PVT reference, at SD acceleration.

At the time of approximately 1.7 seconds, the PVT reference nears the limit of VL[3]= -5000. Before the reference waveform actually reaches -5000, the stop manager finds that the speed is too great for stopping at VL[3] with a deceleration of SD. It therefore reduces the speed, coming to a complete stop at VL[3] with very small or no overshoot.

At approximately 2.75 seconds, the stop manager finds that although the position reference is not yet in range, it *will* be, in a short time. The stop manager begins to accelerate the motor into the permitted range so that catching up when the reference returns to range will be immediate, without the delay of acceleration.

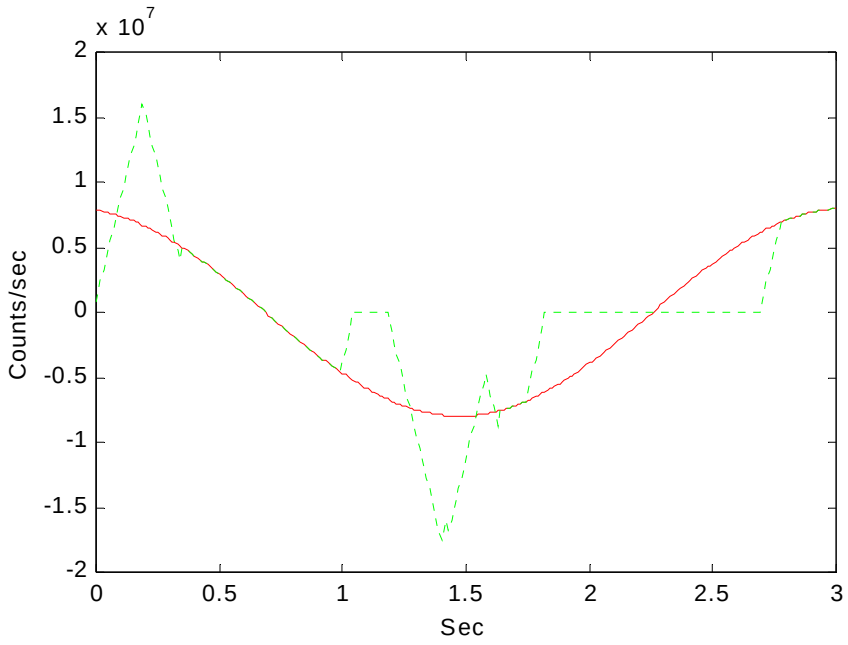
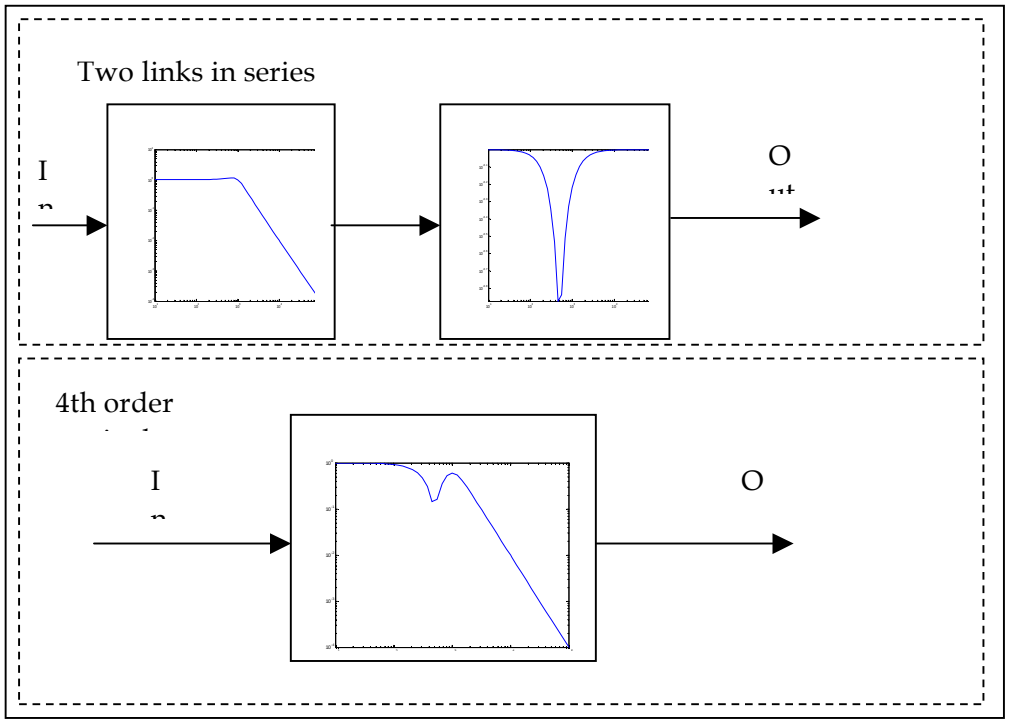


Figure 6-16: **Speed Output of the Stop Manager**

Chapter 7: Filters

The filter serves as a basic building block for the SimpIIQ drive algorithms. The SimpIIQ drive uses a filter mechanism in the following:

- The speed controller high-order filter, which is a control filter placed between the speed PI controller and the torque controller.
- The position controller high-order filter, a control filter placed between the position controller gain and the speed PI controller.
- The analog reference signal to the speed controller, which can be filtered
- Analog sensors may be filtered.



All of these filters are parameterized the same way. Each filter is built by one or more second-order links, connected in a series, as depicted in the following two-block example:

In the top frame, the first block in the series is a second-order low-pass filter; the next one is a notch filter. The bottom frame contains a fourth-order filter resulting from applying the notch filter in series with the low-pass filter.

A first-order filter link can be made by setting some of the second-order filter parameters to zero. The filters are implemented in a normalized form, meaning that the DC gain of each filter (and of the entire filter series) is 1.0.

Very high-order control filters can be used with the *SimpIIQ* drive.

The parameters of all filters are programmed into the vector KV[N], as follows:

Filter	Parameters	Maximum Order
Speed controller high-order filter	KV[0]…KV[21]	
Position controller high-order filter	KV[22]…KV[43]	
Analog position sensor filter	KV[44]…KV[55]	Order 4 (2 blocks)
Analog reference to speed controller	KV[56]…KV[67]	Order 4 (2 blocks

Table 7-1: **KV[N] Filter Parameters**

The high-order filter parameters of the speed controller are as follows:

Parameter	Description	Value Range
KV[0]	Is filter active?	0: Filter not active and bypassed. All other filter parameters ignored. 100: Filter is active.
KV[1]	Last index in KV[N] used for this filter	1+5n, where $1 \le n \le 9$ is the number of used second-order links. The maximum resulting filter order is 18.
KV[2]…KV[6]	Parameters of first link	
KV[7]…KV[11]	Parameters of second link	
KV[12]…KV[16]	Parameters of third link	
KV[17]…KV[21]	Parameters of fourth link	

Table 7-2: **Speed Controller High-order Filter Parameters**

The high-order filter parameters of the position controller are as follows:

Parameter	Description	Value Range
KV[22]	Is filter active?	0: Filter not active and bypassed. All other filter parameters ignored. 100: Filter is active.
KV[23]	Last index in KV[N] used for this filter	23+5n, where $1 \le n \le 5$ is the number of used second-order links. The maximum resulting filter order is 10.
KV[24]…KV[28]	Parameters of first link	
KV[29]…KV[33]	Parameters of second link	
KV[34]…KV[37]	Parameters of third link	
KV[38]…KV[43]	Parameters of fourth link	

Table 7-3: **Position Controller High-order Filter Parameters**

The sensor filter parameters are:

Parameter	Description	Value Range
KV[44]	Is filter active?	0: Filter not active and bypassed. All other filter parameters are ignored. 100: Filter is active.
KV[45]	Last index in KV[N] used for this filter	45+5n, where $1 \le n \le 2$ is the number of used second-order links. The maximum resulting filter order is 4.
KV[46]…KV[50]	Parameters of first link	
KV[51]…KV[55]	Parameters of second link	

Table 7-4: **Sensor Filter Parameters**

The parameters of the analog speed reference filter are:

Parameter	Description	Value Range
KV[56]	Is filter active?	0: Filter not active and bypassed. All other filter parameters are ignored. 100: Filter is active.
KV[57]	Last index in KV[N] used for this filter	57+5n, where $1 \le n \le 2$ is the number of used second-order links. The maximum resulting filter order is 4.
KV[58]…KV[62]	Parameters of first link	
KV[63]…KV[67]	Parameters of second link	

Table 7-5: **Analog Speed Reference Filter Parameters**

7.1. Internal Structure of a Filter Link

There are two types of filter links:

- Type 16, a fixed link for which the next four link parameters are the filter coefficients.

A filter link has five parameters:

Element	Description	Comment
0	Type	16 for fixed links Other: Reserved
1…4	Filter coefficients k_1, k_2, k_3 and k_4	Used only for fixed blocks

The parameters k_1, k_2, k_3 and k_4 are stored inside the *SimpIIQ* drive as integers. As a result, when you read back these coefficients, they may have slightly different values than those that you programmed.

7.1.1 Link Parameterization

The basic continuous-time second-order element is a filter with a unity DC gain:

$$\frac{D}{B}\cdot\frac{Es^2+As+B}{Es^2+Cs+D}$$

Note that this element is very general: It can be used, for example, as a notch filter, a low-pass complex pole filter, a double-lead element or a single pole.

The equivalent discrete form is:

$$\frac{b_0z^2+b_1z+b_2}{z^2+a_1z+a_2}$$

Order	Parameter	Description	Comment
1	k_1	$b_0 + b_1 + b_2$	Float, represented by a long value
2	k_2	$-(b_1 + b_2)$	Float, represented by a long value
3	k_3	$-b_2$	Float, represented by a long value
4	k_4	a_2	Float, represented by a long value

Table 7-6: **Fixed Link Parameters**

The parameter a_1 is obtained explicitly by $a_1 = b_1 + b_1 + b_1 - a_1 - 1$.

7.2. Examples of Filter Implementation

The following examples illustrate how the more common filter links can be implemented. Each example calculates the parameters b_0, b_1, b_2, a_1 and a_2.

Note that the frequency response of a discrete-time filter depends on the sampling time. In the following examples, you should use sampling time $T=2\times TT[1]$.

7.2.1 Low-pass (Complex Pole) Element (Represented by Second-order Block)

The basic continuous-time complex pole element is:

$$\frac{\omega^2}{s^2+2\cdot d\cdot\omega\cdot s+\omega^2}$$

where:

- $\omega = 2\pi\cdot f$ is the angular frequency.
- f [Hz] is the pole frequency.

The discrete equivalent form is:

$$\frac{b_0 z}{z^2 + a_1 z + a_2}$$

where:

- $a_1 = \dfrac{2\omega^2 - \dfrac{8}{T^2}}{\dfrac{4d\omega}{T} + \dfrac{4}{T^2} + \omega^2}$
- $a_2 = \dfrac{\dfrac{4}{T^2} + \omega^2 - \dfrac{4d\omega}{T}}{\dfrac{4d\omega}{T} + \dfrac{4}{T^2} + \omega^2}$
- $b_0 = a_1 + a_2 + 1 = \dfrac{4\omega^2}{\dfrac{4d\omega}{T} + \dfrac{4}{T^2} + \omega^2}$

7.2.2 Notch Filter Element (Represented by Second-order Block)

The basic continuous-time notch filter element is:

$$\frac{\omega_2^2}{\omega_1^2} \frac{s^2 + 2 \cdot d_1 \cdot \omega_1 \cdot s + \omega_1^2}{s^2 + 2 \cdot d_2 \cdot \omega_2 \cdot s + \omega_2^2}$$

where:

- $\omega_1 = 2\pi \cdot f_1$, and f_1 [Hz] is the notch frequency.
- $\omega_2 = 2\pi \cdot f_2$, and f_2 [Hz] is the notch frequency.
- d_1 is the notch damping.
- d_2 is the double-pole damping.

The discrete equivalent form is:

$$\frac{b_0 z^2 + b_1 z + b_2}{z^2 + a_1 z + a_2}$$

where:

- $b_0 = \dfrac{p_1 + c_1 + 1}{q}$, $b_1 = \dfrac{2(1 - p_1)}{q}$, $b_2 = \dfrac{p_1 - c_1 + 1}{q}$, $a_2 = \dfrac{p_2 - c_2 + 1}{q}$
- $a_1 = b_0 + b_1 + b_2 - a_2 - 1 = \dfrac{2(1 - p_2)}{q}$

assuming:

- $c_k = \frac{2d_k}{\tan\left(\frac{\omega_k T}{2}\right)}, p_k = \frac{1}{\tan\left(\frac{\omega_k T}{2}\right)^2}, (k = 1,2)$
- $q = p_2 + c_2 + 1$

7.2.3 Double-lead Element (Represented by Second-order Block)

The basic continuous-time double lead-lag element is:

$$\left(\frac{b}{a} \cdot \frac{s+a}{s+b}\right)^2$$

The frequency $a / (2\pi)$ [Hz] is the lead corner frequency. The frequency $b / (2\pi)$ [Hz] is the lag corner frequency.

The discrete equivalent form is:

$$\left(G + (1 - G)\left(\frac{z(1 - \beta)}{z - \beta}\right)\right)^2$$

Order	Parameter
1	$k_1 = (1 - \beta)^2$
2	$k_2 = G\beta \cdot (G\beta + 2 \cdot (1 - \beta))$
3	$k_3 = - (G\beta)^2$
4	$k_4 = \beta^2$

The DC gain of this block is unity.

β is selected as $\beta = e^{-bt}$.

G is selected by $G = \frac{b}{a}$, with $0 < \beta < 1$ and $1 < G < 3$.

7.2.4 First-order Element (Represented by Second-order Block)

The basic continuous-time single lead-lag element is:

$$\frac{b}{a} \cdot \frac{s+a}{s+b}$$

The discrete equivalent form is:

$$\left(\frac{1-\beta}{1-\alpha}\right) \cdot \left(\frac{z-\alpha}{z-\beta}\right)$$

Order	Parameter
1	$k_1 = 1 - \beta$
2	$k_2 = \left(\frac{1-\beta}{1-\alpha}\right)\alpha$
3	$k_3 = 0$
4	$k_4 = 0$

For a single pole, $\cdot \frac{s}{s+p}$ parameters are $\alpha = 0, \quad \beta = e^{-pT}$.

Example:
A filter consists of one second-order block and one first-order block. The second-order block is a notch filter with a 300-Hz notch frequency and a damping of 0.14. The sample time is 360 microseconds. The discrete equivalent of the filter is parameterized by:
k1 = 0.3372, k2 = 0.4573, k3 = -0.7277 and k4 = 0.5222.

The second block is a simple pole with a 400-Hz frequency. It may be represented by:
k1 = 0.1341, k2 = 0, k3 = 0 and k4 = 0.

The KV parameter vector should be programmed as follows:

Parameter	Value	Description
KV[0]	100	
KV[1]	12	Last index used
KV[2]	16	First block, non-scheduled
KV[3]	IQ25(0.3372)	Parameter 1 (k1)
KV[4]	IQ25 (0.4573)	Parameter 2 (k2)
KV[5]	IQ25 (-0.7277)	Parameter 3 (k3)
KV[6]	IQ25 (0.5222)	Parameter 4 (k4)
KV[7]	16	Second block, non-scheduled
KV[8]	IQ25 (0.1341)	Parameter 1 (k1)
KV[9]	0	Parameter 2 (k2)
KV[10]	0	Parameter 3 (k3)
KV[11]	0	Parameter 4 (k4)

The value of KV[N] parameters for N = 12…47 is not important.

Above, IQ25(x) simply means $round(2^{25}x)$.

Be aware that the KV[N] manual programming is not required because the Composer program performs it manually.

Chapter 8:The Position and Speed Controller

This chapter, which provides details about the speed and position control algorithms, is written for the advanced user who wants to tune the *SimpIIQ* servo drive manually, rather than using the Digital Feedback Technologies tuning systems. Familiarity with basic digital control theory is mandatory.

The type of controller used depends on the *SimpIIQ* drive unit mode

Unit Mode	Control Algorithm
0	Voltage mode
1	Open loop.
2	Speed control.
3	Micro-stepper.
4	Position, dual-feedback source. One sensor is used for commutation and speed control while the other sensor serves for load position control.
5	Position, single-feedback source.

Table 8-1: **Unit Mode Values and Definitions**

At the basic level of the speed controller, the control algorithm is the traditional PI (proportional - integral). At the basic level of the position controller, the control algorithm is an internal PI speed loop and an external position loop using a simple gain for the controller, a structure known as a cascaded loop.

As control requirements become more stringent, the algorithms may extend to include:

- A high-order filter that operates in series with the PI controller, with the following blocks:
  - Notch filters for notching resonance.
  - Low-pass filters for attenuating high-frequency resonance and decreasing sensor noise.
  - A lead-lag element, which may increase bandwidth by adding phase near the crossover frequency.
- A continuous scheduling algorithm that modifies the PI algorithm as a function of the closed loop operating point.

An advanced user may tune the various filter blocks, which is more complex than tuning the PI and simple gain. The gain scheduling can be programmed by the support software automatically or manually, using the Advanced Manual Tuning options.

The following table lists the parameters of the algorithms referred to in this chapter. Details are available in the *SimpIIQ for Steppers Command Reference Manual.*

Parameter	Description
UM	Unit mode. Determines the type of control algorithm used: speed, dual position or open loop.
KP[N]	N=2: Inner speed loop proportional gain N=3: Outer loop gain (UM=4 and UM=5)
KI[N]	N=2: Inner speed loop integral gain, I.
KV[N]	Coefficients for high-order filter. The parameter KV[0] asserts if these filters are used: If KV[0] is zero, the high order filter is not used.
GS[N]	Gain scheduling parameters.
MC	Maximum drive phase current.
WS[28]	Sampling time of the controller, in microseconds.

Table 8-2: **List of All Control Parameters**

Notation:
Standard mathematics notation is used here:

- The time derivative is denoted by the letter s.
- The expression *sx* is equivalent to *dx/dt.*
- The expression *x/s* is equivalent to $\int x\,dt$, where *t* is the time and *x(t)* is any signal.
- The operator *z* denotes a time advance of a single sampling time.

For a digital system with the sampling time of T_s:
$zx(kTx) = x[(k + 1)\ T_s]$ or simply $zx(k) = x(k+1)$.

8.1. Gain Scheduling

Gain scheduling is the process of on-line control algorithm switching, in order to adapt the controller to varying circumstances.

SimpIIQ enables manual or automatic gain scheduling, as set by GS[2].

GS[2] = 0: disables scheduling, the control gains is via KP[N] and KI[N].

GS[2] = 1..16: The controller gains are selected from the KG[N] vector. The selection is manual, according to GS[2]. GS[2]=N selects the N controller out of the bank of 16 controllers. This option is good for scheduling due to load or posture events, that the user program or the host may be aware of.

GS[2] = 64 sets speed-driven automatic gain scheduling, as explained in the following section.

8.1.1 Automatic Gain Scheduling

The most common reason to change the control filter on the fly is a speed change.

When the motor moves very slowly, the encoder count frequency is slow. Because speed estimates are based on encoder differences, it follows that the speed readouts suffer great delays.

An example:

Suppose a control loop with a bandwidth of 300 Hz, and with the motor traveling at 500 count/sec. The average speed delay is $\frac{1}{2\times 500}=1\text{msec}$. At 300 Hz, this costs the phase margin of $300\times 360\times 0.001$=108 deg. As phase margins rarely exceed 50 deg, it is not possible for the controller to remain stable.

The Digital Feedback Technologies tools consider these low-speed delays when producing controller design.

The design result is a battery of 16 controllers; each fits a different time delay, thus fitting another speed. The figure below depicts this.

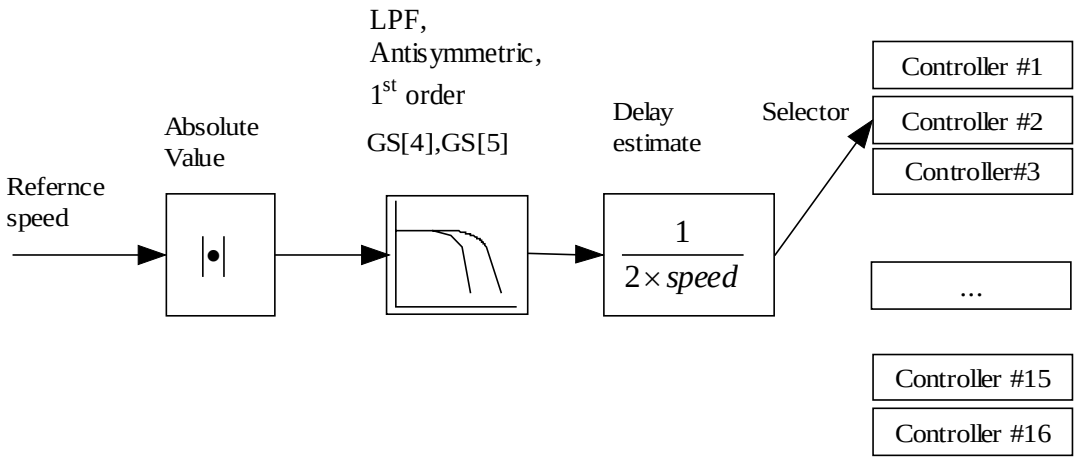


Figure 17: **Gain scheduling**

The absolute reference speed is filtered. The gain scheduling function uses this filtered speed to select the control filter to use from the bank of 16 controllers.

The low-pass filter is not symmetric: GS[4] sets the time constant when the speed demand increases, GS[5] sets the time constant for speed demand decrease.

The separation is since when speed demand increases, we want the controller to swiftly adapt and apply full bandwidth. On the other hand, when speed decreases we allow some time for the actual speed to really be there, and only than reduce gains.

The gain scheduling is derived from the speed reference, not from the speed feedback. This ensures that the gain scheduled control scheme will remain stable.

8.2. Speed Control

8.2.1 Block Diagram

This is the most basic closed loop control form. The basic control block of the speed controller is the PI. The high-order filter (composed of a series of blocks of the types explained previously) and the gain scheduler. A block diagram of the speed controller is given in the following figure.

The units for the speed loop gains are:

KP : mAmp/(count/sec)

KI : mAmp/count

The gain scheduler can be used or the gains of the controller can be fixed using GS[2]=0..15. Use of the high-order filter is optional; to bypass the high-order filter, set KV[0]=0.

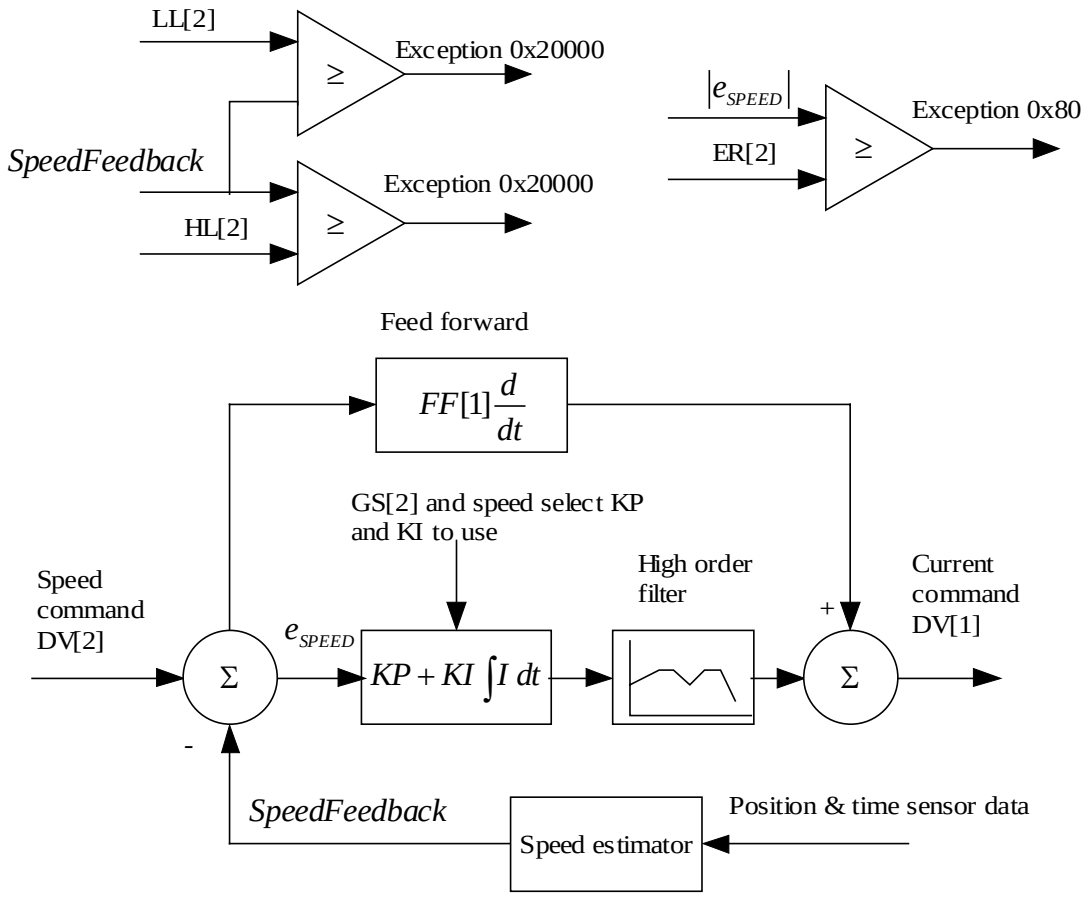


Figure 8-18: **Speed Controller Block Diagram**

The speed controller throws exceptions on excessive speed errors (ER[2]) and on excessive speed (HL[2], LL[2]).

8.2.2 Speed Controller Parameters

The basic continuous-time PI controller is:

$$\frac{K_I + K_P s}{s}$$

where:
K_I is the integral parameter.
K_P is the proportional parameter.

When using the scheduler, K_I and K_P are functions of time. When not using the gain scheduler, they are fixed.

The input to the PI element is the speed error e_{SPEED}(t) [Internal speed units]:

$$e_{SPEED}(t) = SpeedCommand(t) - SpeedFeedback(t)$$

The output to the PI element current command *I(t)* in Ampere units is:

$$I(t) = \int_0^t K_I \cdot e_{SPEED}(\tau) d\tau + K_P \cdot e_{SPEED}(t)$$

The filter has no units and its DC gain is always unity. The filter is described in greater detail in the chapter on Filters.

For a non-scheduled case:
K_P = *KP*[2]
K_I = *KI*[2]

The scheduled case is explained in Section 8.1

The parameters of the non-scheduled speed controller are:

Parameter	Description
KP[2]	Proportional gain
KI[2]	Integral gain
GS[2]	Controller gain selection
FF[1]	Torque feed forward
CL[1]	Continuous torque limit
PL[1]	Peak torque limit
KV[N]	High-order filter parameters

Table 8-3: **Non-scheduled Speed Controller Parameters**

8.3. The Position Controller

8.3.1 Block Diagram

The position controller comprises a proportional gain, cascaded over the speed controller. The block diagram is given in the following figure.

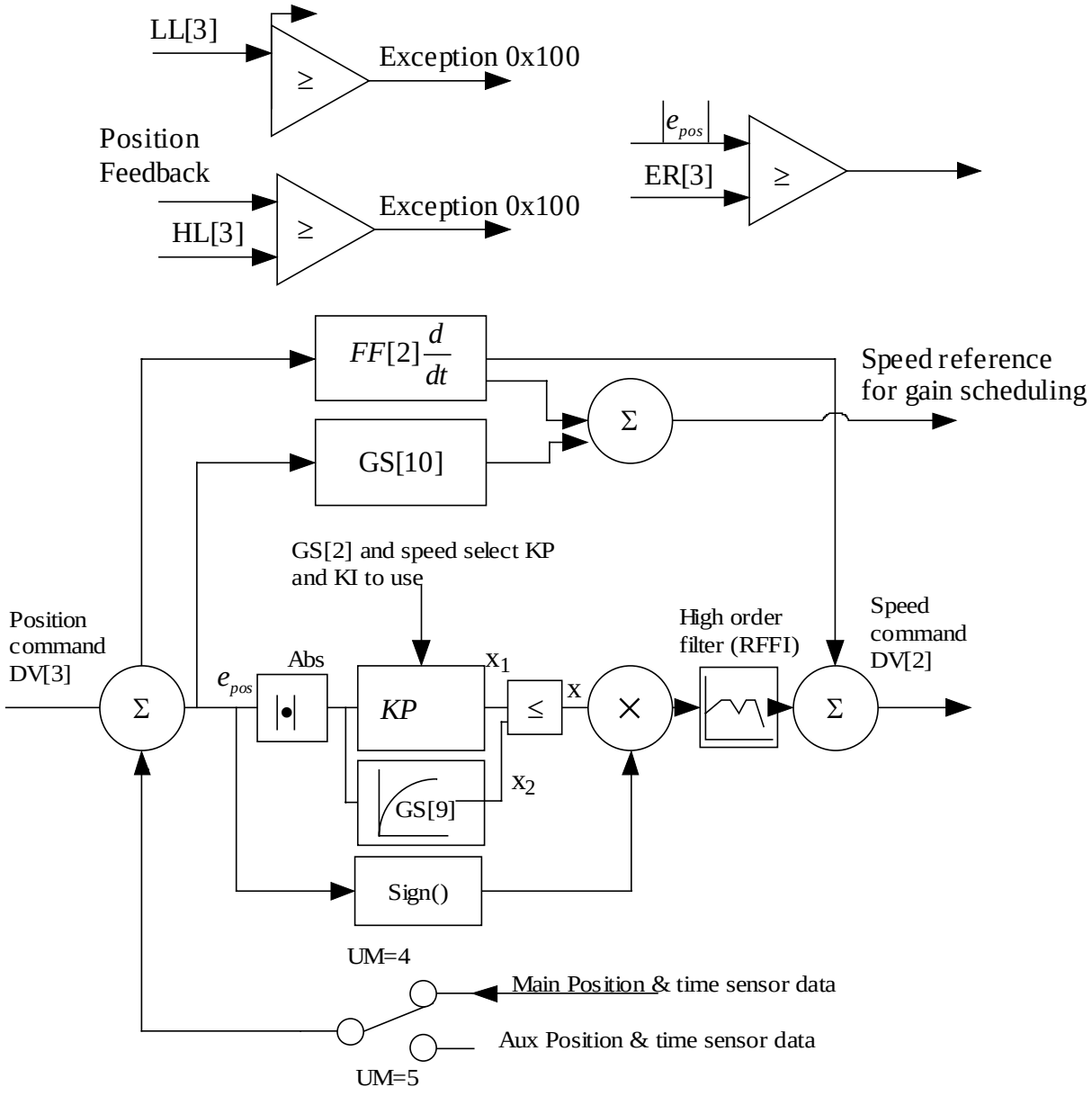


Figure 8-19: **Position Controller Block Diagram**

The reference to the speed controller is composed of the derivative of the position command (speed) and of the output of the position controller. The derivative of the position command is multiplied by the FF[2] . Normally FF[2] matches the units between the position and the speed controllers. For UM=4, FF[2] equals the gear ratio between the speed sensor and the position sensor. For UM=5, the speed feedback is derived from the position sensor, so normally FF[2]=1.

You can use the gain scheduler, or the gains of the controller can be fixed by setting GS[2]=0.

Using the high-order filter is optional. To disable the high-order filter, set KV[48]=0.

The high order filter has no units and its DC gain is always unity. For more details, refer to the chapter on Filters.

The position controller throws exceptions on excessive position errors (ER[3]), and on deviation of the feedback position from its limits LL[3], HL[3].

The gain scheduling is derived from the speed reference + GS[10] x [Position error]. This means that for position control, the gain scheduling does depend on the feedback – otherwise the feedback will become too sluggish at low speeds.

8.3.2 Position Controller Parameters

The position controller drives an inner speed controller.

The position controller is made of a non-linear gain and an optional filter.

Most of the applications use the gain only.

The position controller calculates the position error:

$$e_{POS}(t) = PosCommand(t) - PosFeedback(t) \quad \text{[counts]}$$

then it calculates a linear gain:

$$x_1 = K_p \cdot |e_{POS}|$$

and a non-linear gain:

$$x_2 = \sqrt{2 \cdot GS[9] \cdot |e_{POS}|}$$

and takes $x = \min(x_1, x_2) sign(e_{pos})$

The parameter GS[9] is the maximum deceleration that the motor can drive. This assures, on one hand, that for small position errors the position controller is linear, whereas for larger position errors, the position controller will never require a speed that can't be stopped without overshooting.

The parameters of the non-scheduled position controller are:

Parameter	Description
KP[3]	Proportional gain, K_P^{Out}.
GS[2]	Controller gain selector.
GS[9]	Acceleration limit.
VL[2], VH[2]	Maximum speed command.
VL[3], VH[3]	Maximum position command.
FF[2]	Speed feed-forward.
UM	For UM=4, the position feedback is taken from the auxiliary encoder. For UM=5, the position feedback is taken from the main encoder.
…	All parameters of the speed controller.

Table 8-4: **Non-scheduled Position Controller Parameters**

Chapter 9:The Current Controller

The lowest level controller is the current controller; it is a full vector controller. Its block diagram is in the figure below.

The current mode can be bypassed by direct voltage control, but this is not recommended.

A brushless motor carries alternating currents in its phases. The alternating currents in the motor phases create a rotating magnetic field, which can be projected in two directions. The first magnetic field component (Q) is perpendicular to the magnetic direction of the rotor and produces all the mechanical torque. The second magnetic field component (D) is aligned with the magnetic direction of the rotor; it produces no mechanical torque.

IQ[Amp] is the component of the motor phase currents that creates effective torque. The current controller attempts to make IQ equal to the current command. ID is the component of the motor phase current that does *not* create torque. The current controller tries to null ID.

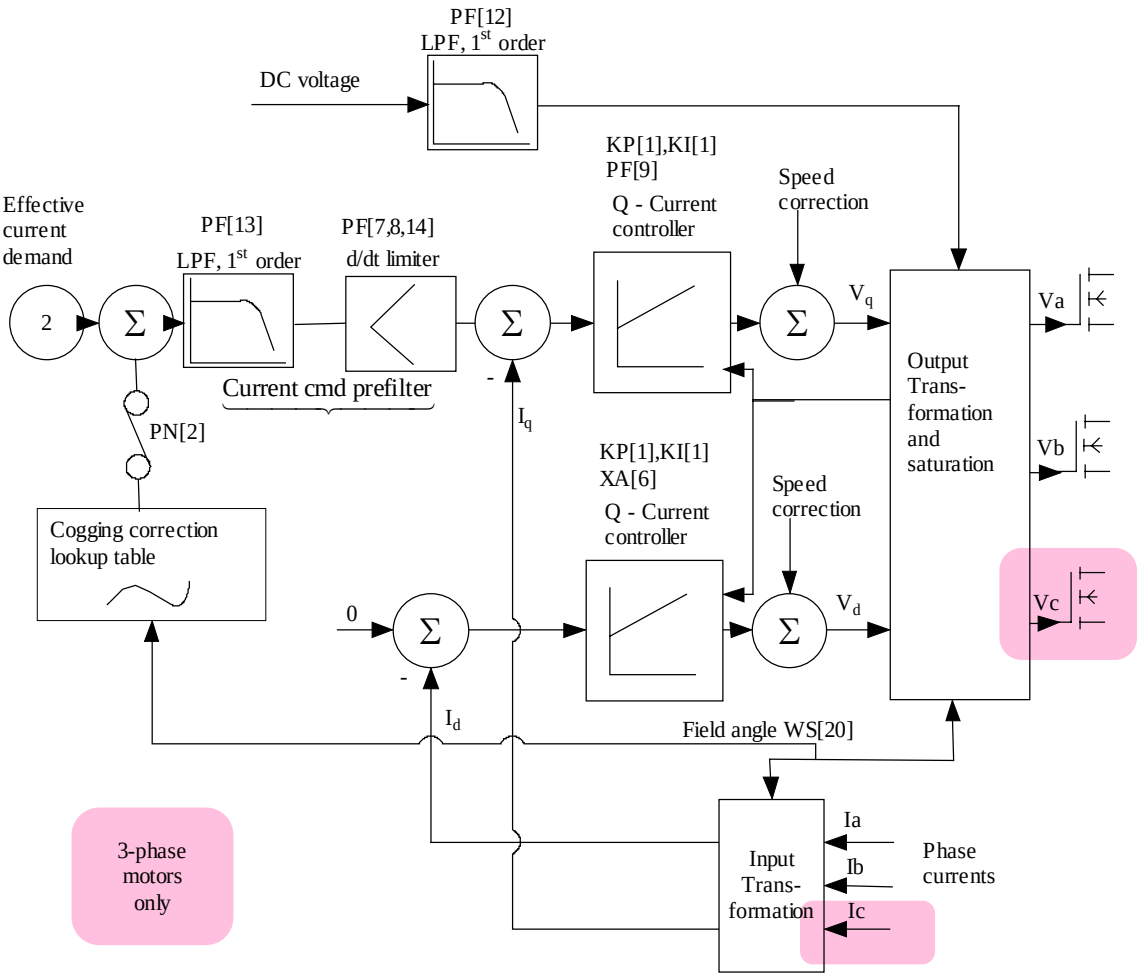


Figure 20: **Block diagram of the current controller**

9.1.1 Input Transformation: Calculating Iq and Id

When the motor winding is sinusoidal, the Iq and the Id are calculated as follows:

$$I_q = \frac{2}{3}\left(I_a \cos(\theta) + I_b \cos(\theta + 120^o) + I_c \cos(\theta + 240^o)\right)$$

$$I_d = \frac{2}{3}\left(I_a \sin(\theta) + I_b \sin(\theta + 120^o) + I_c \sin(\theta + 240^o)\right)$$

(Three phase case, CA[28]=0)

Or

$$I_q = I_a \cos(\theta) + I_b \cos(\theta + 90^o)$$

$$I_d = I_a \sin(\theta) + I_b \sin(\theta + 90^o)$$

(Two phase case, CA[28]=0)

This means that when Iq=1, Id=0 the current through each of the motor windings is sinusoidal, and its peak value is 1 Amp, and its RMS value is 0.707Amp.

SimpIIQ compensates, however, for a non-sinusoidal winding shape. This assures that a 1 Amp effective current demand always produces the same torque, regardless of the motor angle. When winding compensation applies, the winding current waveforms become more complex.

For Iq=1, Id=0 the phase current peak value of 1 Amp is no more guaranteed, but the RMS value of 0.707Amp remains.

9.1.2 Output Transformation: Calculating Phase Voltages

This block transforms the outputs of the Q and D controllers – namely Vq and Vd – to motor phase voltages.

In the simplest sinusoidal, non saturated case, we have

$$V_a = V_q \cos(\theta) + V_d \cos(\theta + 90^o)$$
$$V_b = V_q \sin(\theta) + V_d \sin(\theta + 90^o)$$
$$V_c = -V_a - V_b$$

In the 3-phase case

$$V_a = V_q \cos(\theta) + V_d \cos(\theta + 90^o)$$
$$V_b = -V_a$$
$$V_c = V_q \sin(\theta) + V_d \sin(\theta + 90^o)$$
$$V_d = -V_c$$

In the 2-phase case (stepper)

The math is significantly more complex accounting for winding corrections.

With winding corrections:

- Predicted values are added to the voltages to overcome rotating inductance effects and BEMF.
- Voltages are converted to PWM values, using the available DC supply voltage.
- The neutral point voltage is set.
- Vector saturation is applied as needed.
- If saturation occurred, anti windup corrections are returned to the Q and D controllers.

The final PWM values are sent to the switching transistors.

9.2. Cogging Correction

Cogging is a magnetic friction – it comes from the magnetic asymmetry of the motor. The cogging generates torques or forces that are independent of the motor current.

Because of the cogging, motor motion may be unsmooth. The cogging correction function automatically generates torque additions that compensate for the cogging torque.

Cogging correction works very well at slow speeds. At high speeds, cogging becomes a high frequency disturbance. On the one hand, cogging correction becomes less important because the high frequency less disturbs motion; on the other hand, the correction becomes less efficient due to the limitations of the current amplifier.

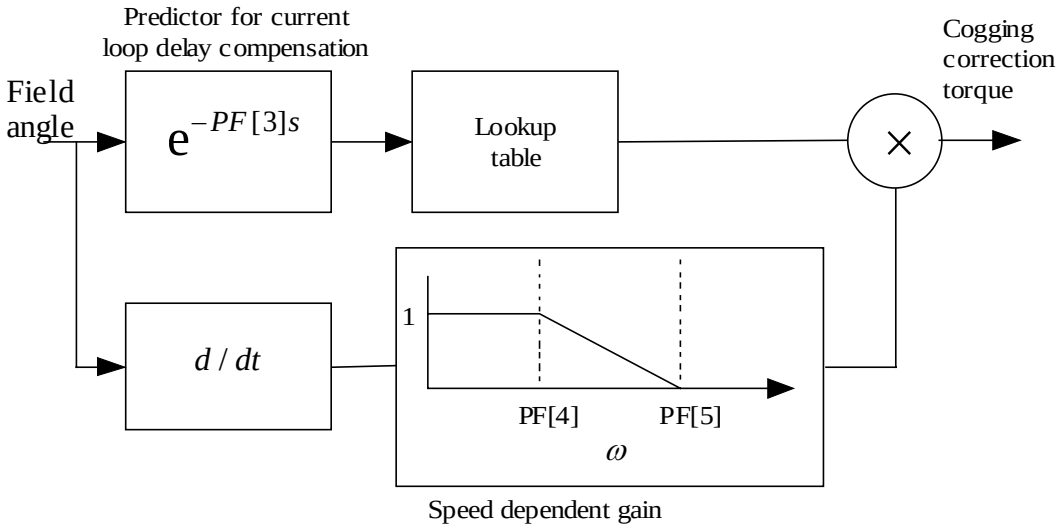


Figure 21: **Cogging correction scheme**

Therefore, *SimpIIQ* uses the cogging correction scheme above in order to apply cogging correction at reasonable speeds, reducing the correction as speed increases.

The cogging correction may be enabled and disabled as a whole, using the parameter PN[2].

9.3. Speed Corrections

The *SimpIIQ* is a full vector drive. It compensates for the following disturbances:

- Winding BEMF shape.
- Voltage required for driving the motor inductance.

All the speed corrections may be disabled or enabled as a whole using the parameter PN[3].

9.3.1 Winding BEMF Shape Correction

The BEMF model serves the following purposes:

- Current scale corrections, in order to minimize ripple torques.
- Winding shape correction, for better current control in the presence of EMF.

The back EMF, for a single motor phase, is modeled by a normalized lookup table, and the motor torque coefficient K_T, given by PF[2].

Both the lookup–tables and K_T are measured by the setup and tuning tools.

9.3.2 Inductor Voltage Corrections

Inductance compensations are required to decouple the D and the Q controllers, i.e. prevent Q channel control effort leaks to the D channel and vice versa.

The correction uses the inductance value in PF[1].

9.4. Current Controller PI + Filter Implementation:

There are two identical current controllers: one for the Q channel and one for the D channel.

Each current controller implements the following block diagram:

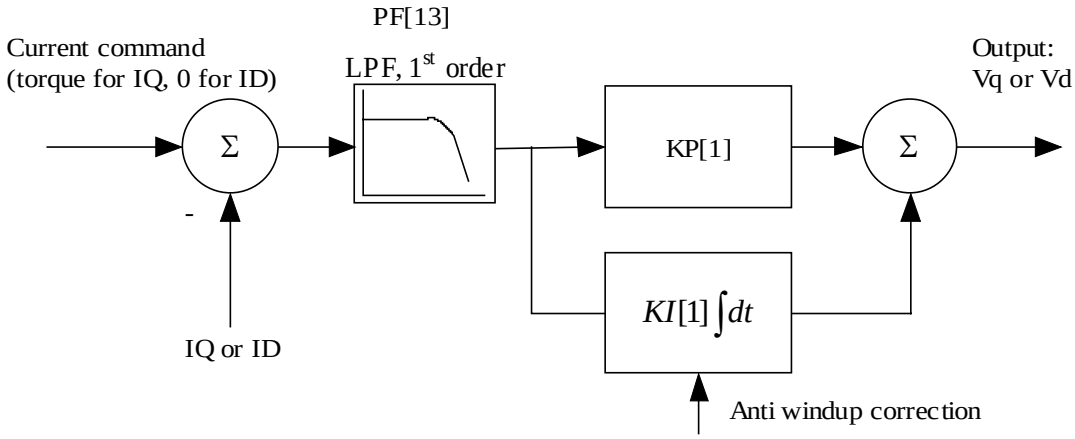


Figure 22: **Current PI controller + filter**

The filter, with its bandwidth PF[13], is for attenuating high frequency noise.

KP[1] and KI[1] are the proportional and integral gains, respectively.

The "Anti Windup correction" means that after limiting the actual controller output (the limiting depends on the output of the other channel and also on the available DC bus voltage), the integrator state is trimmed to avoid excessive overshoot that voltage limits may cause.

The units of KP[1] are Volt/Amp

The units of KI[1] are Volt/(Amp Sec)

Normally, the Digital Feedback Technologies tuning tools set the values for KP[1], KI[1], and PF[13].

9.5. Pre-filter and Limiter

This block prevents excessive current overshooting.

The elements of this block are:

PF[13] sets a 1st order low-pass filter at the current demand input.

PF[14] limits the current command increase in one sampling time.

PF[7] and PF[8] together set a dynamic limit of the current command rate, to prevent overshooting the device-rated current, even when the current command jumps its full range in an instance.

Chapter 10:Development Aids

SimpIIQ hardware and software include a number of features that facilitate application development:

- The *SimpIIQ* drive's built-in simulation capability enables the user to develop large parts of the application from the desktop, without even connecting the *SimpIIQ* drive to a motor.
- Controller sampling time can be optimized for best performance.
- The recorder is a major tool for viewing signals and debugging an application.

10.1. The Recorder

The *SimpIIQ* drive recorder enables the user to record up to eight signals simultaneously. The recorded signals can be uploaded to the host through the communication connection, for presentation and analysis. *SimpIIQ* drives operating with CAN have two communication lines: CAN and RS-232. This arrangement enables the recorder to use one line for performance monitoring while the machine host uses the other line to control the *SimpIIQ* drive in its normal context.

This section explains how to define the recorder parameters for the *SimpIIQ* drive, how to launch and trigger the recorder and how to fetch the recorded data.


Under most circumstances, this command is used only by the Composer or by the tuning environment.


For implementations with separate motion processor, the *SimpIIQ* drive and the motion processor have a separate, independent recorder for each. The AR[1] command selects which recorder is addressed by the recorder commands.

The list of recordable signals supported by the *SimpIIQ* drive is stored internally and can be retrieved by the host. (Refer to the LS and LP commands in the *SimpIIQ for Steppers Command Reference Manual*.) In addition, the *SimpIIQ* drive can record user-program variables for tracing the progress of user programs without interrupting them.

The following commands are relevant to the recording process:

Command	Description
BG, BT, IL[N]	Begins motion using software or hardware command. Start of motion may be used to trigger the recorder.
BH	Uploads recorder results.

Command	Description
LS	Loads a record from the serial flash memory, in order to retrieve the list of recorder signals.
RC	Defines which of mapped signals should be recorded.
RG	Recorder gap: specifies sampling rate of the recorder.
RL	Recorder length.
RP[N]	Recorder parameters: defines which event will trigger the recorder, and the trigger position. Also defines basic time quantum for recorder (TS or four times TS) and data to be uploaded to host by next BH command.
RR	Launches recorder and reads back its status.
RV	Signal mapping: maps the IDs of the recordable signals to logical IDs that the recorder can reference.
SR	Status register, which indicates the status of the recorder: idle, armed, triggered and recording, or ready with data.
TS	Sampling time, the basic resolution of recorder.
WI[21]	Actual amount of recorded data.

Table 10-1: **Commands Relevant to the Recorder**

10.1.1 Recorder Sequencing: Programming, Launching and Uploading Data

In order to activate the recorder, it must first be programmed to indicate which signals should be recorded, at what resolution and what will trigger the recording event. A number of limitations apply to programming the recorder:

- The recorder cannot be programmed while it is armed or recording. It must first be stopped, or "killed" (RR=-0).
- Modifying the recorder programming invalidates any previously recorded data. Be sure to upload all data that you need before programming the recorder.

After it is programmed, the recorder can be launched by:

- RR=1 Arms recorder to trigger at next start of motion.
- RR=2 Starts recording immediately.
- RR=3 Arms recorder to start recording at next trigger event.

To poll the status of the recorder, use the RR and SR commands. After the recorder data is ready, use the BH command to upload the data.

10.1.2 Signal Mapping

The recorder can record a range of different signals. The first 16 signals — listed in the following table — are compatible with other Elmo drives.

Signal ID	Signal Name (Command)	Length; Type	Description
1	Main speed (VX)	Long Float	Speed of main feedback sensor, in counts/second.
2	Main position (PX)	Long Integer	Cumulative position of main feedback sensor, in counts.
3	Position command (DV[3])	Long Integer	Position reference, in counts. When external reference mode is set (RM=1), this signal also includes analog reference command.
4	Digital input pin status	Short Integer	IP variable (see IP in the *SimpIIQ for Steppers Command Reference Manual*).
5	Position error (PE)	Long Integer	Position tracking error: the difference between main position reference and main position feedback, in counts.
6	Current command (DV[1])	Short Float	Current reference to current controller, in amperes.
7	DC bus voltage	Short Integer	Bus voltage, in volts.
8	Auxiliary position (PY)	Long Integer	Cumulative position of auxiliary feedback sensor, in counts.
9	Auxiliary Speed (VY)	Long Float	Speed of auxiliary feedback sensor, in counts/second.
10	Active current command (IQ)	Short Float	Active part of vectored current reference, in amperes.
11	Reactive current command (ID)	Short Float	Reactive part of vectored current reference, in amperes.
12	Analog input 1 (AN[1])	Short Float	Analog input 1 after offset compensation of AS[1], in volts.
13	Reserved		
14	Motor current phase A (AN[3])	Short Float	Phase A current, in amperes.
15	Motor current phase B (AN[4])	Short Float	Phase B current, in amperes.
16	Speed command (DV[2])	Long Float	Speed reference to speed controller, in counts/second.

Table 10-2: **Default Mapping of Recorded Signals**

After power on, the recorder can access the first 16 signals, as listed in Table 10-2. To access other signals, the recorder must perform a process called "mapping" in order to make them available. In the mapping process, the IDs of the desired signals are mapped to recorder "cells" (logical IDs that can be directly referenced by the recorder) so that on power on, the signals are mapped to the cells. The following table lists the additional signals that are available to the *SimpIIQ* drive recorder. The full list of signals may differ according to the *SimpIIQ* drive's grades and released. The signals can be retrieved from the personality partition of the serial flash memory.

Signal ID	Signal Name (Command)	Length; Type	Description
17	Field angle	Short	Stator field angle, 1024 counts per electrical revolution.
18	Commutation sensor	Long Integer	Position counter, counted modulo per mechanical revolution, with origin at electrical angle of zero.
21	Filtered torque command	Short	Command to Q current controller, at output of command filter.
45	Motor DC supply voltage	Short	Sample motor DC supply voltage.
64	External position reference	Long Integer	Part of position reference generated by external inputs.

Table 10-3: **Additional Recorded Signals**

Prior to being recorded, the signals listed in Table 10-3 must be mapped to the recorder. Up to sixteen signals may be mapped to the recorder at any time. Up to eight signals can be recorded simultaneously.

The RV[N]=x command maps a signal with the ID of x to the logical ID of N, N=1…16. RC is a bit-field parameter (bit 0 to 15) that sends the actual required signal to the recorder. In this case, bit N-1 of RC points to signal x.

Example:
RV[2]=1 maps the signal with the ID of 1 (main encoder speed) to the logical ID of 2. This means that in order to record the main encoder speed, bit 1 of RC should be set (RC=2).

The default mapping, restored at boot, maps the signals of IDs 1 through 16 to the corresponding logical IDs 1 through 16. Therefore, the signals with IDs 1 through 16 can be recorded using corresponding logical IDs, without further programming.

No mapping is required if only signals with IDs 1 through 16 are needed.

10.1.3 Defining the Set of Recording Signals

The RC command defines which mapped signals should be recorded. Each bit of RC, a 16-bit bit field, specifies a logical signal ID for recording. For example, if RC=5, the first and third bits with the binary value of five will be on and the rest of the bits will be zero (0000000000000101). Therefore, RC=5 specifies that the signals with logical IDs one and three should be recorded, and all other signals should not.

Example:
The commands `RV[1]=5; RV[2]=1; RC=3;`
define that when launched, the recorder will record the main speed and position error.

RC may define a maximum of eight recorded signals, meaning that the binary representation of RC may not include more than eight ones.

10.1.4 Programming Length and Resolution

The RP[0], RL and RG commands are used to program the length and resolution of the recorded signals.

RP[0] defines the basic time quantum of the recorder:

- RP[0]=0 synchronizes the recorder to the speed or position control cycles. The time quantum of the recorder is 4 * TS.
- RP[0]=1 synchronizes the recorder to the torque cycles. The time quantum of the recorder is TS.

RG defines the sampling rate of the recorder, in terms of the time quantum:

- RG=1 indicates that a new sample should be taken once per time quantum. If TS=70 and RP[0]=0, the recorder will sample once per 280 microseconds.
- RG=2 indicates that a new sample should be taken once per two time quanta. If TS=70 and RP[0]=1, the recorder will sample once per 140 microseconds.
- Similarly, RG=N indicates the a new sample should be taken once per N time quanta.

Note that RG specifies only the recorder sampling rate, not the trigger sampling rate. A trigger event will cause the recorder to start within one time quantum (TS).

RL defines the maximum number of data items to collect. For example, if RL=11, TS=70, RG=1 and RP[0]=0, then the 11 samples will be taken at the rate of 280 microseconds, lasting (11-1) x 280 milliseconds = 2,800 microseconds.

The recorder memory is limited to a total of 4096 long variables. When more signals are recorded, less memory is available for each recorded signal.
If RL > (recorder memory) / (number of signals), the recorder will fail to record the specified RL samples and instead, will record up to its data volume.

The actual amount of recorded data can be polled using the WI[21] command.

10.1.5 Trigger Events and Timing

The recorder is started by a trigger event, which is one of the following:

- Immediate: The recorder starts immediately after the recording request has been issued.
- Begin motion: The recorder is triggered by the execution of a software BG command, by a timed BG command, or by a hardware command.
- Analog signal: The recorder starts upon one of the following:
  - Positive slope: The signal crosses a prescribed level with a positive slope.
  - Negative slope: The signal crosses a prescribed level with a negative slope.
  - Window: The signal exits a window of two prescribed signal levels.

  The following graph depicts the analog signal trigger types:

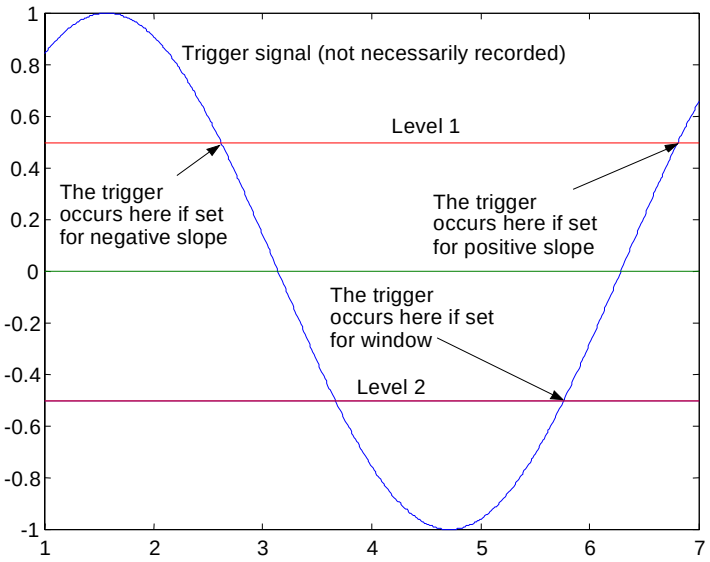


Figure 10-1: **Slope and Window Trigger Types**

- Digital signal: The *SimpIIQ* drives will support this trigger option in the future.

The trigger defines when the recorder is to start. The recorder can be programmed to start *before* the trigger event, so that the trigger event can occur "in the middle of the action." This is possible because the recorder begins to record at the instant it is launched by the RR command, so that when the trigger event occurs, the pre-trigger information is already recorded.

Example:
The signal to be recorded is speed, and the trigger is set on BG. After launching the recorder with `RR=3`, the following command sequence is given: `JV=5000;BG`.

The following graph depicts the pre-trigger delay:

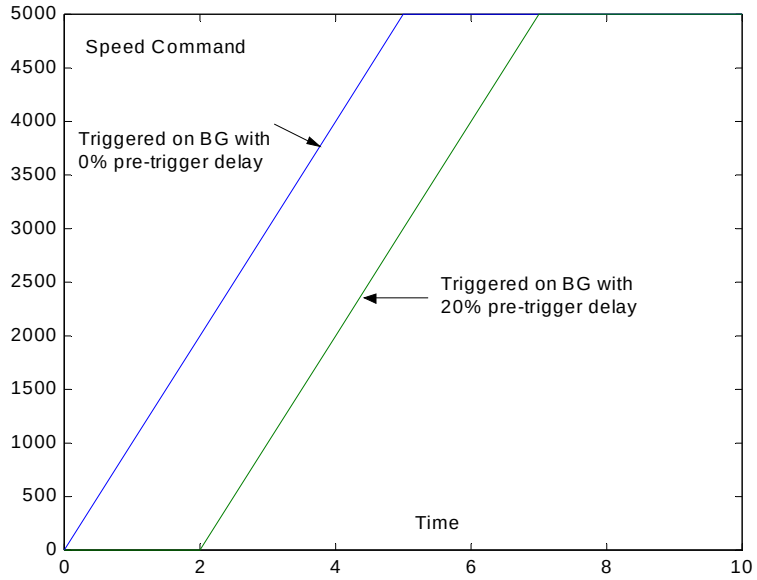


Figure 10-2: **Pre-trigger Delay**

In this example, the recorder works for 10 seconds. In such a case, a pre-trigger delay of 20% requires 2 seconds, in order to acquire the pre-trigger data. A `BG` command set for less than 2 seconds after the `RR=3` will be missed.

The following table lists the trigger parameters.

RP[N]	Description	Definition
RP[0]	Recorder time quantum	0: Speed controller sampling time (4 * TS µsec) 1: Torque controller sampling time (TS µsec)
RP[1]	Trigger variable	Similar to RC, but only 1 bit may be non-zero. The trigger variable does not need to be a recorded variable. For example, RV[1]=17, RP[1]=1 defines a trigger that is made on signal #17 (stator field angle).
RP[2]	Pre-trigger storage, in percent	0 to 100 (percent).
RP[3]	Trigger type	0: Immediate 1: BG 2: Positive slope 3: Negative slope 4: Window 5: Trigger on digital input
RP[4]	Level 1	Level of positive slope trigger, or high side of window trigger.
RP[5]	Level 2	Level of negative slope trigger, or low side of window trigger.
RP[6]		Digital input trigger polarity.

RP[N]	Description	Definition
RP[7]		Digital input trigger mask

Table 10-4: **Trigger Parameters**

Trigger levels RP[4] and RP[5] can be entered as either integers or floating point numbers.

10.1.6 Launching the Recorder

The RR command is used to launch (or kill) the recorder. It also reports the recorder status:

- RR=-1 kills the recorder if active, and invalidates any recorded data.
- RR=0 kills the recorder and does nothing if it is not active.
- RR=1 launches the recorder, triggered on next BG.
- RR=2 launches the recorder with an immediate trigger.
- RR=3 launches the recorder with the trigger defined by the RP parameters.

Note that the results of RR=1 and RR=2 can be achieved with RR=3 and the appropriate RP[N] definitions. The RR=2 and RR=2 commands are used for easy interface with the most common recorder actions.

As a status report, RR may return the following values:

- RR=-1: No valid data in recorder.
- RR=0: Recorder action is complete and is loaded with valid data.
- RR=1, 2 and 3: Waiting for the completion of RR=1, RR=2 or RR=3 respectively.

  The report value (1, 2 or 3) does not indicate if the recorder is already recording or if it is waiting for a trigger. If this differentiation is required, use the SR command.

The SR (status register) command details the status of the recorder. SR returns a bit field, in which bits 16 and 17 may have the following values:

Bit 16	Bit 17	Recorder Status
0	0	Recorder inactive, no valid recorded data
0	1	Recorder armed, waiting for trigger event
1	0	Recorder finished, valid data ready for use
1	1	Recording now

Table 10-5: **SR Recorder Status Reports**

10.1.7 Uploading Recorded Data

The following commands are used to upload recorded data from the drive to a host:

Parameter	Description
RR	If zero, indicates that the recorder is ready for data upload
WI[21]	Indicates how much data is actually recorded
RP[8], RP[9]	Defines the part of the signal to be uploaded next
BH	Uploads recorded data command

Table 10-6: **Parameters for Uploading Recorded Data**

The BH command is used to upload to the host the values recorded by the *SimpIIQ* drive recorder. BH optimizes the data transfer, assuming that the host has the computing power to analyze the *SimpIIQ* drive message.

To execute a BH command, valid data must be stored in the recorder. If the data is valid, the fields of the RC variable define the variables that have been recorded.

The BH=n command uploads the recorded variable defined by RC&n, where & is the bit-wise AND operator. For example, if RC=7, the command BH=2 will transfer the variable to be recorded by setting RC=2, because 7&2=2. If the binary representation of RC&BH includes more than one "1", the variable with the lowest value will be uploaded and BH will not return an error.

Example:
If `RC=7` and `BH=3`, `BH` will upload the recorder variable defined by `RC=2`. `BH=16` will return an error because `BH&RC=0`.

It is convenient to use hexadecimal notation for the BH command; for example, BH=0x4000 is more readily understandable than BH=32768.

The BH command can load an entire recorded signal, or a part of one. If RP[8]=0 and RP[9]=0, BH=n will upload an entire signal. Otherwise, BH will upload the recorded signal from index RP[8] until index RP[9]. RP[9] must always be less than or equal to the length of the recorded signal.

The data is uploaded in hexadecimal form in order to minimize transmission time (relative to ASCII formatted text), while adhering to the ASCII nature of transmissions. Each data byte is parsed into two nibbles, and the ASCII code of the nibbles is sent from the controller to the host. For example, the short integer number 43794 has the hexadecimal representation AB12. It will be transmitted as "A" "B" "1" "2" with the most significant nibble first and the least significant nibble last. The long integer number 1 will be sent as: "0" "0" "0" "0" "0" "0" "0" "1".

In order to analyze the BH record, it is important to understand that the internal representation of quantities in the controller is not in user units. For example, while the user relates to motor current in amperes, the controller represents current internally by the bits of the A/D that measures the current. The BH record uploaded recorded currents (and other variables as well) in their internal representation units, and also provides the scaling multiplier to relate it to user units. This way, no multiplication inaccuracies are introduced to the BH records, and the CPU load is minimized.

The recorded transmitted by the controller in response to a BH=n command is described in the following table. The record includes 20 bytes of overhead, and numerical record data. The variables in the table are translated into ASCII as described previously.

Byte Number	Description	Value	Type
0 - 1	Variable type for user. Field has no practical significance.	0: Integer system parameter 1: Real system parameter 2: Integer user program variable 3: Real user program variable	Byte
2 - 3	Data width: number of hex character of single transmitted data item.	4: Short integer 8: Long integer	Byte
4 - 7	Data length: actual number of transmitted data items.		Word
8 - 11	Variable time multiplier: number by which TS must be multiplied to obtain basic recording period.	Depends on RP[0] value	Word
12 - 19	Floating number factor, by which every uploaded data item is multiplied in order to convert it to user units, such as amperes, or counts/second.		32-bit float number in IEEE format.
20 - 20+	(Data length) * (Data width) -1: data items. Oldest record is transmitted first, and most recent record is transmitted last.		Words or long integers, according to data width.

Table 10-7: **BH Record Structure**

BH record transmission time can be quite long. A record of 2000 long numbers is approximately 8000 bytes, which take at least 4 seconds to transmit over RS-232 at a baud rate of 19,200. During this time:

- The user program continues to run normally.
- CAN commands are accepted and processed normally.

- RS-232 commands are accepted and executed normally, although the transmission of the response to them is deferred until after the BH upload is complete.

Example:
A BH command may return the string:
`0008000100013f80000000010000`

This string is decomposed to a field as follows:

`00` (int) `08`	8 bytes per data item in the message
`0001`	Only one data item in message, after the 20-byte overhead
`0001`	Record taken every TS
`3f800000`	To scale, multiply result by 1.0. The floating point IEEE representation of 1.0 is 0x3f800000.
`00010000`	First data item: 0x10000 = 65536

Chapter 11:Miscellaneous Topics

11.1. Index Identification for Analog Encoders

Analog encoders normally come with an analog index signal. For example, refer to the analog waveform in the diagram below. Because the index is made by comparing the analog index signal to zero volts, it is wide, and will latch different positions for CW and for CCW approaches.

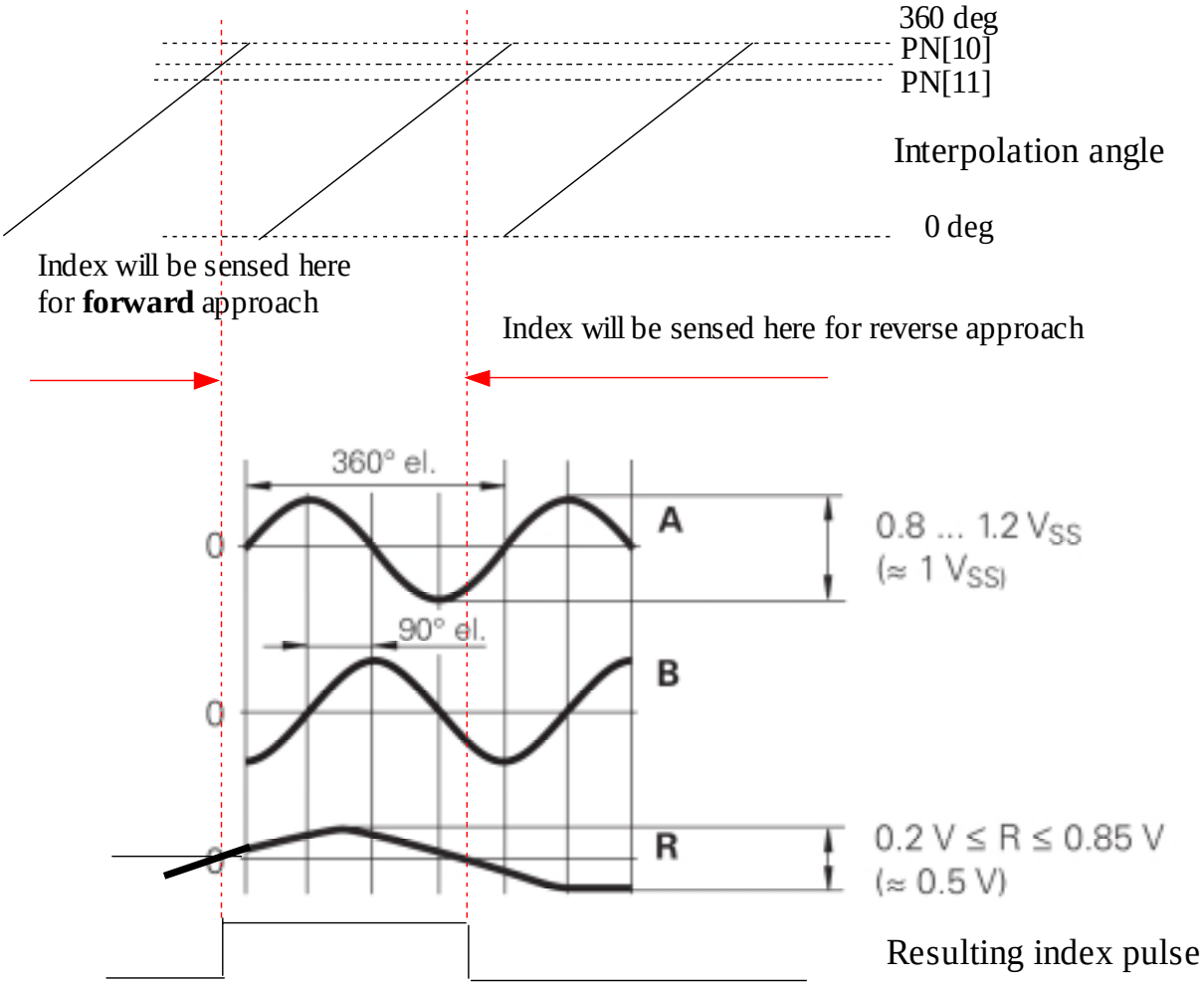


Figure 11-1: ***Analog encoder signals, based on the Heidenhain RON 486 datasheet***

For the index to be fully repetitive, SimplIQ needs to know the electrical angle that corresponds to each index pulse edge; these are given by the parameters PN[10] and PN[11], respectively. The units for PN[10] and PN[11] are 65536 for 360 degrees.

Notes:

- The tuning wizard in the Composer program has a utility to find and set PN[10] and PN[11] correctly.
- The analog index will appear to work correctly in most cases even if PN[10] and PN[11] had never been tuned. There is however a danger: if either the PN[10] value or PN[11] value deviate about 180° from their correct setting, the index may be identified with the error of one full encoder line (360°), and consecutive homing processes may yield inconsistent results.