INFRANOR
SERVO DRIVES & MOTION CONTROL

Preliminary

Gem Drive
User Guide

gb

WARNING

This is a general manual describing a series of servo drives having output capability suitable for driving AC brushless sinusoidal servo motors.
Please see Gem Drive Installation Guide for the hardware installation of the drive (dimensions, wiring, ...).
Instructions for storage, use after storage, commissioning as well as all technical details require the MANDATORY reading of the manual before getting the drives operational.
Maintenance procedures should be attempted only by highly skilled technicians having good knowledge of electronics and servo systems with variable speed (EN 60204-1 standard) and using proper test equipment.
The compliance with the standards and the "CE" approval is only valid if the items are installed according to the recommendations of the drive manuals. Connections are the user's responsibility if recommendations and drawings requirements are not met.

Any contact with electrical parts, even after power down, may involve severe physical damage.
Wait for at least 5 minutes after power down before handling the drives (a residual voltage of several hundreds of volts may remain during a few minutes).

ESD INFORMATION (ElectroStatic Discharge)
INFRANOR drives are conceived to be best protected against electrostatic discharges. However, some components are particularly sensitive and may be damaged if the drives are not properly stored and handled.

STORAGE
The drives must be stored in their original package.
When taken out of their package, they must be stored positioned on one of their flat metal surfaces and on a dissipating or electrostatically neutral support.
Avoid any contact between the drive connectors and material with electrostatic potential (plastic film, polyester, carpet…).

HANDLING
If no protection equipment is available (dissipating shoes or bracelets), the drives must be handled via their metal housing.
Never get in contact with the connectors.

ELIMINATION
In order to comply with the 2002/96/EC directive of the European Parliament and of the Council of 27 January 2003 on waste electrical and electronic equipment (WEEE), all INFRANOR devices have got a sticker symbolizing a crossed-out wheel dustbin as shown in Appendix IV of the 2002/96/EC Directive.
This symbol indicates that INFRANOR devices must be eliminated by selective disposal and not with standard waste.

INFRANOR does not assume any responsibility for physical or material damage due to improper handling or wrong descriptions of the ordered items.
Any intervention on the items, which is not specified in the manual, will immediately cancel the warranty.
Infranor reserves the right to change any information contained in this manual without notice.

Issue: 0.9

GENERAL DESCRIPTION

1. INTRODUCTION

Gem Drive all-digital drives with sinusoidal PWM control are servo drives that provide the control of brushless AC motors with position sensor (resolver, encoder).

Gem Drive is a configurable, programmable drive with built-in servo-functions.

Gem Drive drive is a single-axis block including power supply unit and mains filters. It is available in both 230 VAC and 400/480 VAC mains operated voltages.

It obtains high regulation performances with:

- 16 bits resolver processing
- Multi-standard encoder input
- Motor speed up to 25000 rpm.

It includes an SD card interface for the update of firmware and files.

It also includes a CAN and RS232 communication port.

Gem Drive drives communication is based by default on "CANopen" protocol and drive profile DS402.
The drive parametrization can be made by means of:

- The specific parametrization software **Gem Drive Studio** via the serial port RS-232 or the **CANopen** bus or
- Commands from a controller via the **CANopen** bus.

This manual is supposed to be used by people who have to define a controller configuration to communicate with drives and by people who have to write programs. Most of the internal variables of the drive can be accessed by using the fieldbus or the setting tools. These variables are listed in an “embedded dictionary” including the variable name and properties. All parameters of a drive are contained in a dictionary depending on the firmware version.

In this manual, we will use the generic and standard vocabulary to describe these variables:

The variables are specified as “parameters” from the communication side.

Each parameter is identified by:

an **Index** number
a Sub-index number
a Name

Each parameter has the following properties:

Access type: it is possible to **r**ead it , to **w**rite it….; "**ro**" ” means "read only" , “**rw**“ means "read & write".
Length: byte, word (16 bit), long (32 bit).
Unit
Save type
Possibility or not to access the parameter by using fast communication CANopen services (**P**rocess **D**ata **O**bject service **PDO).** If yes, the field “PDO mapping” of the object dictionary will be “yes”.

Convention: A numerical field can be filled in with numerical values described as “hexadecimal” or “decimal”. An hexadecimal value will be written “ 0x*value*”.

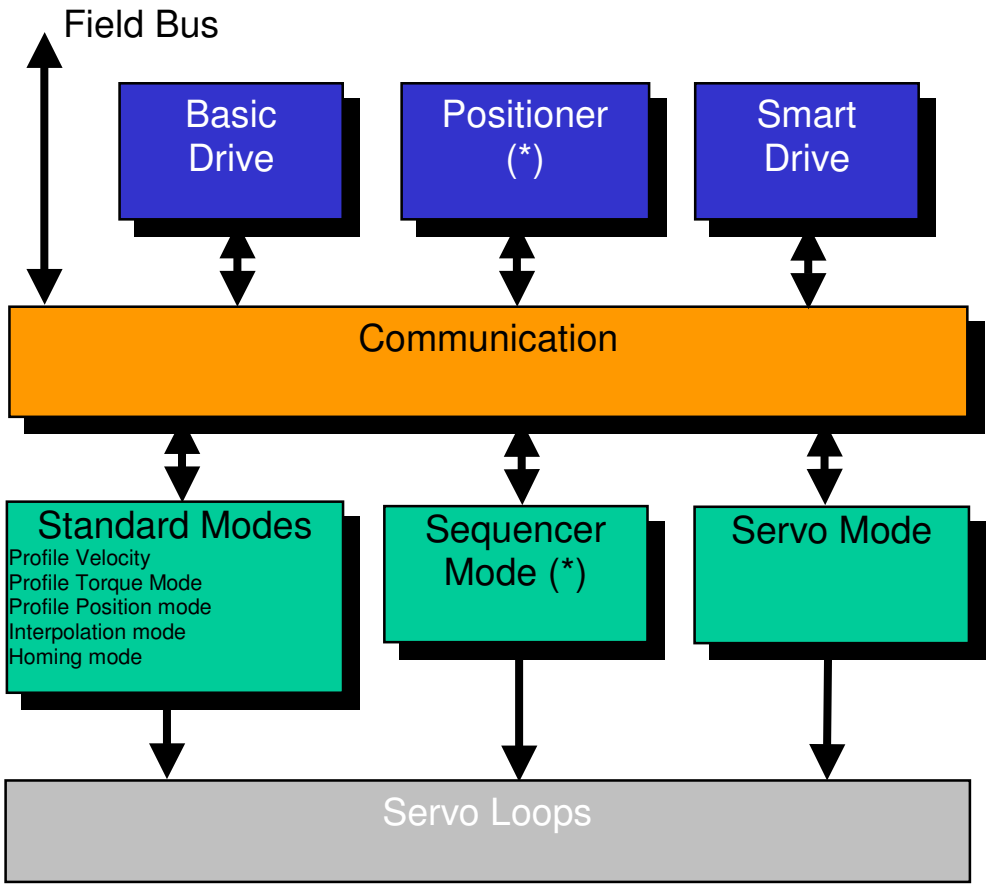
2. ARCHITECTURE OF THE DRIVE

2.1. Functional architecture of the Gem Drive

Gem Drive is a free configurable and programmable drive.
The configuration can be made by static parameters set during the commissioning, by using “Gem Drive Studio”, using a configuration file “USERPARAM.TXT” or commands from a master.
The Gem Drive configuration includes servo-loop parameters, motor and sensor parameters, communication parameters and I/O configuration parameters.
Gem Drive can be used as a “basic drive” using an analog input as a speed set point or CANopen command according to DS402 and additional proprietary features.
Gem Drive can be used as a “positioner" which can be controlled via digital I/Os or field bus and which is able to proceed Position, Speed, Torque command and automatic execution of “motion sequences” (in preparation).
Gem Drive can be used as a “smart drive”. When this operation mode is selected, the behaviour of the drive has to be programmed by using the programmable task.
The motion generation mode can also be selected:

- Among standard modes (Position, Speed, Torque, Homing, Interpolated Position),
- By running the built-in sequencer which automatically runs the “motion sequences” (in preparation),
- By using the powerful concept of servo mode which allows to define the position setpoint as a combination of data, processed by the programmable task and/or the built-in servo functions and/or data issued from the fieldbus.

The following figure describes the functional architecture of Gem Drive



2.2. Definitions

2.2.1. STANDARD MODES

The standard modes are extracted from the CANopen Drive Profile DSP-402.

Each mode can be managed by:

- The device control state machine allowing to control all drives
- The operation modes (Profile velocity, position, etc…)
- All parameters associated to each mode (speed, position, acceleration , etc…).

2.2.2. SEQUENCER MODE (IN PREPARATION)

This mode allows the execution of various sequence types and also the sequence chaining.
Sequences consist of standard mode parameters as previously described.

Sequences supported:

HOME	*Homing Sequence*
ABSOLUTE	*Absolute Positioning Sequence*
RELATIVE	*Relative Positioning Sequence*
SPEED	*Speed Sequence*
TORQUE	*Torque Sequence*

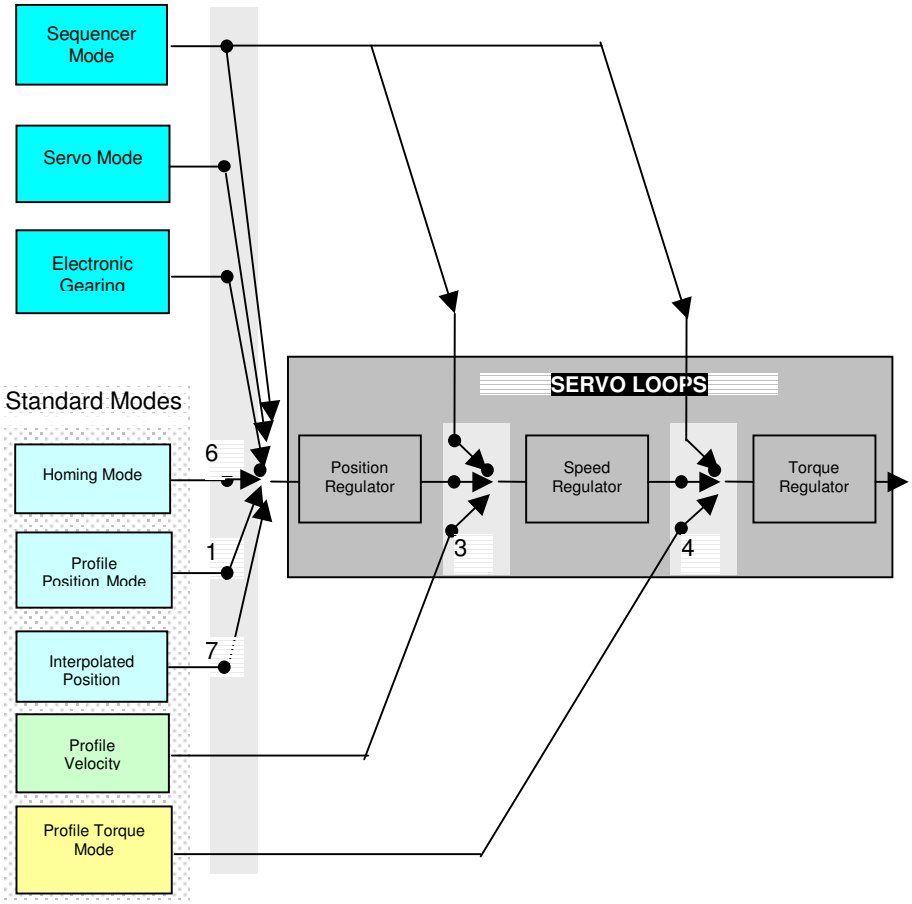
2.2.3. SERVO MODE

The Servo mode allows to easily combine various motion functions in order to perform complex motion applications.

The motion functions can be standard functions, position sequences or complex functions as Gear Boxes, Camming.

2.3. Motion mode structure

The motion mode selection is exclusive. The mode switching can be made when the drive is enabled.



2.4. SERVO MODE

2.4.1. CONCEPT

When activated, the servo mode allows the combination of several signals to generate a position loop set point. The data which can be combined are issued from the network. They are the result of internal calculation made in the “user program” or are output of built-in servo functions.

The built-in servo functions are defined as versatile functions for which input/output/parameters can be defined by the user.

The available servo functions are described in detail in the pertaining section of this manual.

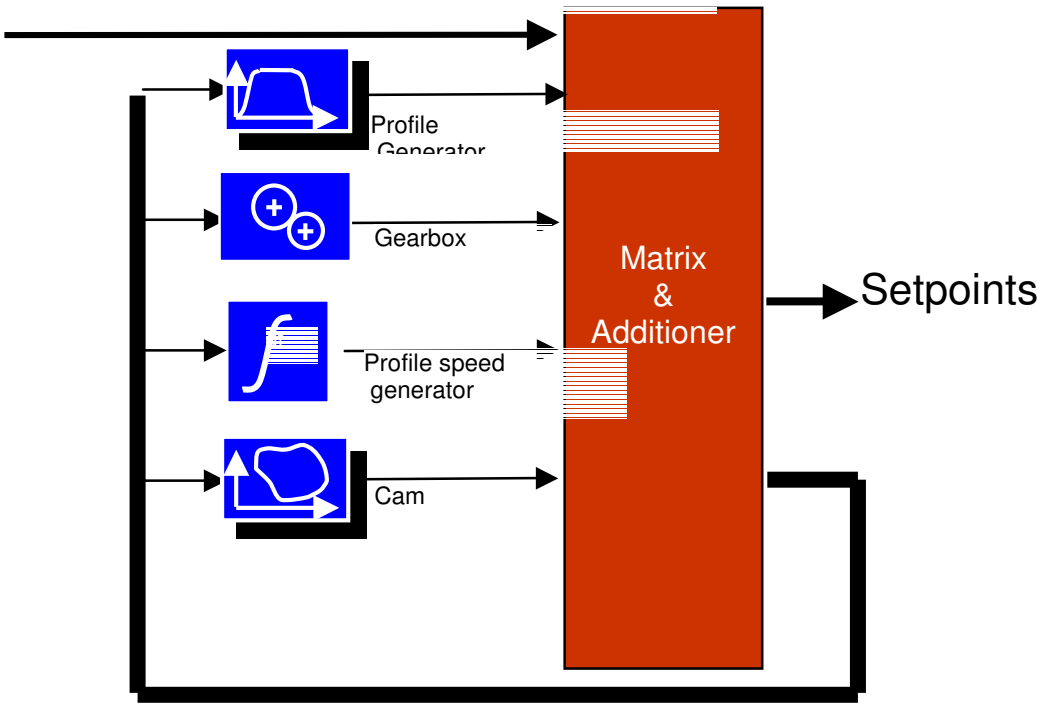
The data combination is performed through a “connexion matrix” which establishes a link between the inputs and evaluates the result of the combination.

The various “routines” which are selected when the servo mode is selected, are executed just before the position loop calculation with the same sampling time.

Three built-in servo functions can be evaluated in the same cycle.

2.4.2. DIAGRAM

Data Flow (CAN, Analog inputs, Internal signals…)



2.5. TOOL KIT

In addition to the servo functions, **Gem Drive** offers several tools that help the user to realize accurate and fast applications:

- In-line interpolator which allows to interpolate data issued from the user program or the fieldbus according to the synchro period of the application and the internal sampling time of the position loop.
- Gem Drive is able to manage the **Synch** signal for CANopen. This feature allows to realize coordinated applications without master.
- Gem Drive offers digital cams and capture features which help to get accurate reaction or positions, whatever the fieldbus communication speed.

2.6. PROGRAMMING

Three task can be programmed:

- a background task dedicated to the initialization of the variables, complex or long calculation, not real time operations.
- A user cyclic task which sampling period is defined by the user.
- A fast cyclic task at same sampling time as the position loop .

All three tasks are programmable via Gem Drive Studio. The programming language is based on IEC 1131.3 Instruction List . Some restrictions are defined for a stable execution of cyclic tasks.

Gem Drive Studio includes an editor and a compiler for program creation, tools for downloading and starting the program in the drive and monitoring tools for an easy debugging.

COMMISSIONING

CAUTION !
Do not make the drive parametrization by means of both Gem Drive Studio and CANopen bus at the same time.
INSTALLATION OF Gem Drive Studio
The Gem Drive Studio software is PC compliant under Windows®[1] and allows an easy parameter setting of the Gem Drive.
Please see our website www.infranor.fr for downloading the "Gem Drive Studio" software.
CHECKING THE DRIVE HARDWARE CONFIGURATION
The standard drive configuration is adjusted to MAVILOR motors (resolver sensor with transformation ratio = 0.5).
For the adjustment to other motor types, please see "Gem Drive - Installation Guide".
POWERING OF THE DRIVE
Please see manual "Gem Drive - Installation Guide" before switching on the drive for the first time.
For switching on the drive, please proceed as follows:

- Switch on the +24V auxiliary supply:

The red front panel LED "ERR" must be flashing ("Undervolt." error displayed).
The AOK relay contact is closed. It is then possible to control the Power ON relay.

- Switch on the power supply:
  The red LED "**ERR**" must be switched off: the drive is ready to be enabled.

CAUTION !

The 24 V auxiliary supply must **always** be switched on **before** the power supply.

[1] Windows® is a registered trade mark of MICROSOFT® CORPORATION

1. STARTING AND ADJUSTING THE DRIVE

This chapter describes the commissioning procedure of the drive by means of the " Gem Drive Studio" software. Gem Drive Studio is a project oriented software. The user has to create at first a project in which he declares the various Gem Drives which are connected together, using CANopen. For each added drive, the user will be asked to give a name and the node ID which is coded on the front panel. He will also have to select an “embedded dictionary file” and then the dictionary which was downloaded together with the software package.
It is mandatory to select the node ID and the bus speed in a consistent way: different addresses, same speed.

- Connect the serial link RS232 between PC and drives.
- Switch on the drives and start the **Gem Drive Studio** software on the PC.
- Configure the communication by selecting the com port that will be used on the PC, as well as the address range of the connected drives.
- Connect the PC to the drives network. The “bullet” in front of each node must be green.

At this stage, Gem Drive Studio checks that the actual “embedded dictionary” is complying with the drive’s one.
It is now possible to read the “embedded dictionary of the drive” (Tools > Read an embedded dictionary in a drive).
Selecting the device info window, when connected, allows the software to recognize the characteristics of the drives.

1.1. DRIVE ADJUSTMENT TO THE MOTOR SPECIFICATIONS

1.1.1. CONFIGURATION OF THE SENSORS

All internal setpoints and displays are given by using the “user unit“ definition. At this stage, it will be necessary to define the sensors and also the relationship between sensors data and “user unit value”. The drive is configured for a resolver as motor sensor. For applications requiring a position encoder:

- Select the appropriate encoder type in the Position Sensor menu.
- Adjust the scaling factors
- Then select Encoder feedback and confirm this selection.

Perform the Save parameters procedure before switching off the drive in order to save the sensor configuration.

1.1.2. SELECTION OF THE MOTOR TYPE

- **THE MOTOR USED IN THE APPLICATION IS CONTAINED IN THE MOTOR LIST OF THE PARAMETRIZATION SOFTWARE**

Select, in the motor list, the motor used in the application.

The motor selection will start the automatic calculation of the current loop parameters.

Check that the values of the parameters **Max. current** and **Rated current** are compliant with motor and drive. If necessary, modify them according to the motor and drive specifications.

The parameter **Max. current** defines the maximum output current value of the drive. It may vary between 20 % and 100 % of the drive current rating.

The parameter **Rated current** defines the limitation threshold of the drive output RMS current (I^2t).

It can vary between 20 % and 50 % of the drive current rating.

If the resolver phasing is unknown, the motor phasing can be launched either in the control window of the **Gem Drive Studio** software or via the CANopen bus.

- **THE MOTOR USED IN THE APPLICATION IS NOT CONTAINED IN THE MOTOR LIST OF THE PARAMETER SETTING SOFTWARE**

Select the **New Motor** function and follow the instructions.

1.1.3. CONFIGURATION OF THE THERMAL SENSOR

The thermal sensor is entering both X1 (resolver) and X3 (encoder) connectors according to the motor position feedback sensor.

Selection of the sensor type

The motor can be equipped either with a CTN sensor (ohmic resistance = decreasing temperature function) or with a CTP sensor (ohmic resistance = increasing temperature function).

Check that the selected thermal sensor type actually corresponds to the sensor type mounted on the application motor.

Triggering threshold adjustment

Enter the sensor ohmic value (kOhm) corresponding to the required temperature value for the release of the **Motor over_temperature** protection, according to the manufacturer's specifications.

Warning threshold adjustment

Enter the sensor ohmic value (kOhm) corresponding to a warning temperature value.

When the warning temperature is reached, an information is sent via the **CANopen** bus.

Note

When using a CTN sensor, the warning ohmic value will be higher than or equal to the triggering ohmic value.

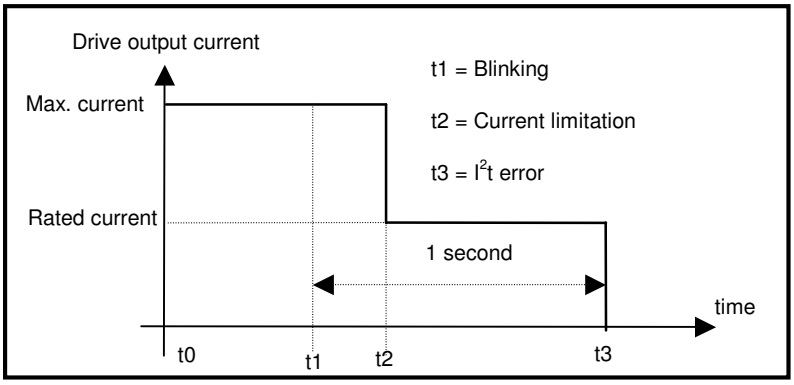
When using a CTP sensor, the warning ohmic value will be lower than or equal to the triggering ohmic value.

1.1.4. MOTOR RATED CURRENT I²T

2 selection modes are available: Fusing or Limiting.
It is advisable to use the Fusing mode during commissioning phases.
In Fusing mode, the drive is disabled when the current limitation threshold is reached.
In Limiting mode, the motor current is only limited at the value defined by the Rated current parameter when the limitation threshold is reached.

- **OPERATION OF THE CURRENT LIMITATION IN "FUSING" MODE**

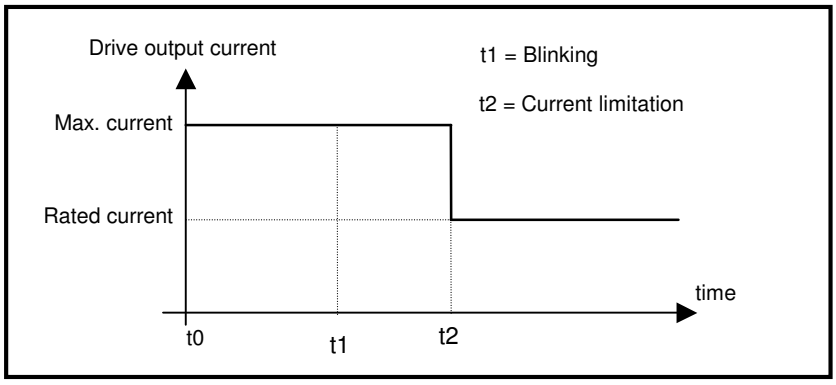
When the drive output RMS current (I^2t) reaches 85 % of the rated current, the OK LED is flashing on the drive front panel. If the RMS current (I^2t) has not dropped below 85 % of the rated current within 1 second, the I^2t error is released and the drive disabled (otherwise, the flashing is inhibited).
When the drive output RMS current (I^2t) reaches the rated current value, the I^2t limits the drive output current at this value.
Diagram of the drive output current limitation in an extreme case (motor overload or shaft locked):



The maximum current duration before release of the blinking display is depending on the value of the parameters Rated current and Max. current. This value is calculated as follows:
T_{dyn} (second) = t_1-t_0 = 3,3 x [rated current (A) / max. current (A)]^2
The maximum current duration before limitation at the rated current is also depending on the value of the Rated current and Maximum current parameters. This value is calculated as follows:
T_{max} (second) = t_2-t_0 = 4 x [rated current (A) / max. current (A)]^2
NOTE 1
When the "Max. current / Rated current" ratio is close to 1, the Tdyn and Tmax values given by the formula above are quite below the real values. But this formula remains very precise as long as the "Max. current / Rated current" ratio is higher than 3/2.
NOTE 2
The drive I^2t signal can be displayed on the digital oscilloscope by selecting the I^2t signal in the Channel menu.
The threshold values of the I^2t signal, for the protection mode described above, are calculated as follows:
Triggering threshold of the Idyn signal (%) = [Rated current (%)]^2 / 70
Current limitation threshold (%) = [Rated current (%)]^2 / 50
Rated current (%) = 100 x Rated current (A) / drive current rating (A)
The corresponding RMS current value of the drive can be calculated as follows:
Drive RMS current (A) = [I^2t signal value (%) x 50]$^{1/2}$ x drive current rating (A) / 100

- **CURRENT LIMITATION IN "LIMITING" MODE**

When the drive output RMS current (I^2t) reaches 85 % of the rated current, the OK LED on the drive front panel is flashing. When the RMS current (I^2t) drops below 85 % of the rated current, the flashing is inhibited.
When the drive output RMS current (I^2t) reaches the rated current value, the I^2t protection limits the drive output current at this value.
Diagram of the drive output current limitation in an extreme case (motor overload or shaft locked):



The maximum current duration before release of the flashing display (t1 - t0) and before limitation at the rated current (t2 - t0) is calculated the same way as in the "Fusing" mode.

2. - SERVO LOOP ADJUSTMENT

2.1. REGULATOR PARAMETERS

The Autotuning procedure identifies the motor and load specifications and calculates the speed/position loop parameters.
In velocity mode, only the speed loop gains are calculated.
In Position mode, all gains of both speed and position regulators are calculated.
The operator can select a bandwidth (Low, Medium or High) as well as the filter type (standard, antiresonance or max. stiffness).
The Autotuning procedure can be executed with the motor disabled or enabled (i.e. vertical load).
Before executing the Autotuning procedure, check that the motor shaft is free and that its rotation over one revolution is not dangerous for operator and machine. Check that the brake is released (the Autotuning command does not control the brake).
For a complete adjustment, the Autotuning procedure must always be executed in Position mode.

- check that the motor is correctly running in both directions,
- check the response at a small displacement without Idc saturation (oscilloscope function).

In case of loud noise in the motor at standstill or when running, check the rigidity of the mechanical transmission between motor and load (backlashes and elasticities in motor and couplings).
If required, start a new Autotuning procedure by selecting a lower bandwidth.
If the instability remains, start a new Autotuning procedure by activating the Antiresonance filter. If necessary, adjust more accurately the loop response stability by adjusting the stability gain.
If the Autotuning procedure was executed in PI^2 mode, when the Position mode is selected, the Feedforward gains of the position regulator must be adjusted manually. Set the Feedforward speed 1 gain value at 1, in order to avoid a high following error value.

2.2. LOOP ADJUSTMENT WITH A VERTICAL LOAD

In the case of an axis with vertical load, proceed as follows:
Select the Limiting current limitation mode.
Initialize the speed loop gains corresponding to the unloaded motor (execute therefore the Autotuning procedure with the motor uncoupled from its mechanical load).
Couple the motor with its load. If possible, make a control in speed mode; otherwise, close the position loop with a stable gain.
Select the PI^2 speed mode and move the axis by means of the speed input command until a stall position where one motor revolution is not dangerous for operator and machine (far enough from the mechanical stops).
Execute then the Autotuning procedure with the motor at standstill. If the axis is moving, the Autotuning procedure is not accepted by the drive.
Select the Position mode and set the Feedforward speed 1 gain value at 1, in order to avoid a high following error value.

2.3. ROTATION / COUNTING DIRECTION

The counting direction can be reversed by selecting the Reverse movement in the Gem Drive Studio software.

2.4. PARAMETER SAVING

When all adjustments have been made, the parameters may have to be stored in the flashprom memory of the drive using the "save parameters" command.
It is also useful to get a copy of the drive parameters file for service and diagnostics.

FUNCTIONAL FEATURES

1. LOGIC INPUTS

Gem Drive offers the use of built-in functions for the drive operation. These functions are available using “logical signal” or digital input. The default values of the parameters are set to “logical”.
If required, it will be necessary to “connect” a digital input as a signal activating the function. Details to realize this operation are included in the I/Os section of Chapter ”References".

1.1. "ENABLE" INPUT

Activating this function allows the drive to provide torque on the motor according to the selected motion mode and control-word value.

Desactivating the ENABLE input during the operation makes the axis decelerate. At the end of the deceleration, the drive and the “Motor brake" output are automatically disabled.
Please pay attention to the fact that for consistency between logical signal and electrical signal, when a digital input is used as “ENABLE INPUT”, it is strongly recommended to use a 24 Vdc signal on the input to “enable” the drive. This means that the corresponding digital input has to be “connected” to the corresponding “logical signal” and its polarity has to be set at 1 (please refer to Chapter 5 "References" for details).
Note:
The deceleration can be chosen as ramp or current Deceleration. The corresponding parameters can be set via the field bus or Gem Drive Studio software.

1.2. “ CAPTURE INPUT”

The Capture function allows to record motor position and/or second sensor measurement when an external signal changes.

1.3. "INDEX" INPUT

In Homing mode, according to the machine structure, it may be necessary to connect a digital sensor to identify the real position of an axis. In this case, a digital I/O has to be connected to this function.
Index input is also a possible input for the capture function.

2. BRAKE CONTROL

The Gem Drive has got a control for the operation of a "powerless" brake.
The brake control is enabled (relay open) or disabled (relay closed) according to the drive status (enabled or disabled). See Chapter "References" for details about timings.

3. ADDRESSING SWITCH / SPEED SELECTION

Each drive of the network must be configured with one single address.
A DIP8 switch accessible to the operator allows the configuration of the drive address as well as of the CANopen bus communication speed.

- **Addressing (6 selection bits)**

Status of the cursors						Address
6	**5**	**4**	**3**	**2**	**1**	
OFF	OFF	OFF	OFF	OFF	OFF	0
OFF	OFF	OFF	OFF	OFF	ON	1
OFF	OFF	OFF	OFF	ON	OFF	2
...	...	...	...	...	...	...
ON	ON	ON	ON	ON	ON	63

- Communication speed (2 selection bits)

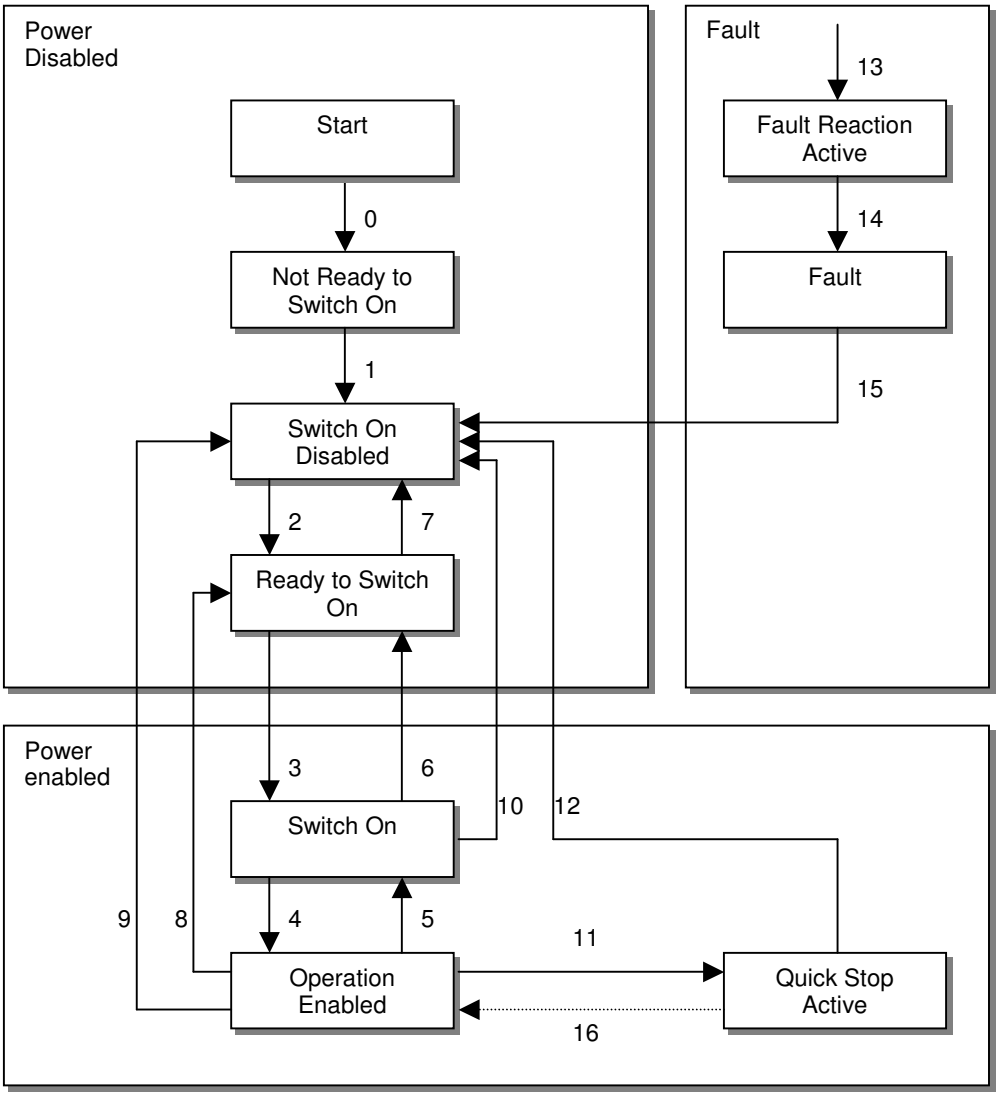
Status of the cursors		Speed
8	**7**	
OFF	OFF	1Mbits
OFF	ON	500Kbits
ON	OFF	250Kbits
ON	ON	reserved

REFERENCES

1. DEVICE PROFILE

1.1. DRIVE STATE MACHINE

The state machine describes the status and the control sequence of the drive.



DRIVE STATES:

The following states of the device are possible:

• NOT READY TO SWITCH ON
Low level power has been applied to the drive.
The drive is being initialized or is running self test.
A brake, if available, has to be activated in this state.
The drive function is disabled.

• SWITCH ON DISABLED
Drive initialization is complete.
The drive parameters have been set up.
Drive parameters may be changed.
High voltage may not be applied to the drive, (e.g. for safety reasons).
The drive function is disabled.

• READY TO SWITCH ON
High voltage may be applied to the drive.
The drive parameters may be changed.
The drive function is disabled.

• SWITCHED ON
High voltage has been applied to the drive.
The power amplifier is ready.
The drive parameters may be changed.
The drive function is disabled.

• OPERATION ENABLED
No faults have been detected.
The drive function is enabled and power is applied to the motor.
The drive parameters may be changed.
(this corresponds to normal operation of the drive.)

• QUICK STOP ACTIVE
The drive parameters may be changed.
The quick stop function is being executed.
The drive function is enabled and power is applied to the motor.

• FAULT REACTION ACTIVE
The drive parameters may be changed.
A fault has occurred in the drive.
The quick stop function is being executed.
The drive function is enabled and power is applied to the motor.

• FAULT
The drive parameters may be changed.
A fault has occurred in the drive.
High voltage switch-on/-off depends on the application.
The drive function is disabled.

STATE TRANSITIONS:

State transitions are caused by internal events in the drive or by commands from the host via the *control word*.

- **State Transition 0**: START -> NOT READY TO SWITCH ON Event: Reset. Action: The drive is self-testing and/or self-initializing.
- **State Transition 1**: NOT READY TO SWITCH ON -> SWITCH ON DISABLED Event: The drive has self-tested and/or initialized successfully. Action: Activates communication.
- **State Transition 2:** SWITCH ON DISABLED -> READY TO SWITCH ON *Event: 'Shutdown' command received from host. Action: None*
- **State Transition 3**: READY TO SWITCH ON -> SWITCHED ON Event: 'Switch On' command received from host. Action: The power section is switched on if it is not already on.
- **State Transition 4**: SWITCHED ON -> OPERATION ENABLE Event: 'Enable Operation' command received from host. Action: The drive function is enabled.
- **State Transition 5**: OPERATION ENABLE -> SWITCHED ON Event: 'Disable Operation' command received from host. Action: The drive operation will be disabled.
- **State Transition 6**: SWITCHED ON -> READY TO SWITCH ON Event: 'Shutdown' command received from host. Action: The power section is switched off.
- **State Transition 7**: READY TO SWITCH ON -> SWITCH ON DISABLED .

Event: 'Quick Stop' and 'Disable Voltage' command received from host.
Action: None

- **State Transition 8**: OPERATION ENABLE -> READY TO SWITCH ON

Event: 'Shutdown' command received from host.
Action: The power section is switched off immediately, and the motor is freely rotating if unbraked.

- **State Transition 9**: OPERATION ENABLE -> SWITCH ON DISABLED

Event: 'Disable Voltage' command received from host.
Action: The power section is switched off immediately, and the motor is freely rotating if unbraked.

- **State Transition 10**: SWITCHED ON -> SWITCH ON DISABLED

Event: 'Disable Voltage' or 'Quick Stop' command received from host.
Action: The power section is switched off immediately, and the motor is freely rotating if unbraked.

- **State Transition 11**: OPERATION ENABLE -> QUICK STOP ACTIVE Event: 'Quick Stop' command received from host. Action: The quick stop function is executed.
- **State Transition 12**: QUICK STOP ACTIVE -> SWITCH ON DISABLED

Event: 'Quick Stop' is completed or 'Disable Voltage' command received from host.
This transition is possible, if the Quick-Stop-Option-Code is different from 5 (stays in the state 'Quick Stop Active').Action: The power section is switched off.

- **State Transition 13**: All states -> FAULT REACTION ACTIVE A fault has occurred in the drive. Action: Execute appropriate fault reaction.
- **State Transition 14**: FAULT REACTION ACTIVE -> FAULT

Event: The fault reaction is completed.
Action: The drive function is disabled. The power section may be switched off.

- **State Transition 15**: FAULT -> SWITCH ON DISABLED

Event: 'Fault Reset' command received from host.
Action: A reset of the fault condition is carried out if no fault is currently present in the drive.
After leaving the state Fault, the Bit 'Fault Reset' of the control word has to be cleared by the host.

• **State Transition 16**: QUICK STOP ACTIVE -> OPERATION ENABLED
Event: 'Enable Operation' command received from host. This transition is possible if the Quick-Stop-Option-Code is 5, 6, 7 or 8.
Action: The drive function is enabled.

1.2. Object definition

0x6040	Control Word
0x6041	Status Word
0x605A	Quick Stop option code
0x6060	Mode of Operation
0x6061	Mode of Operation Display

1.2.1. CONTROL WORD

Index	**0x6040**
Name	Control Word
Object Code	VAR
Data Type	Unsigned16
Object Class	all
Access	rw
PDO Mapping	Possible
Default Value	0000

Bit Number	**Function**
0	Switch On
1	Disable Voltage
2	Quick Stop
3	Enable Operation
4	Operation Mode Specific
5	Operation Mode Specific
6	Operation Mode Specific
7	Reset Fault (rising edge)

Device control commands are triggered by the following bit patterns in the control word:

Command / Bit of the control_word	**bit 7 Fault Reset**	**bit 3 Enable Operation**	**bit 2 Quick Stop**	**bit 1 Disable Voltage**	**bit 0 Switch On**	**Transition**
Shutdown	X	X	1	1	0	2, 6, 8
Switch On	X	X	1	1	1	3
Disable Voltage	X	X	X	0	X	7, 9, 10, 12
Quick Stop	X	X	0	1	X	7, 10, 11
Disable Operation	X	0	1	1	1	5
Enable Operation	X	1	1	1	1	4, 16
Fault Reset	↑	X	X	X	X	15

Bit 4, 5, 6 are operation mode specific:

Mode	**Bit 4**	**Bit 5**	**Bit6**
Profile Position Mode	new set point	change_set_immediately	0: absolute 1: relative
Homing Mode	Homing Operation Start	reserved	reserved
Interpolated Position Mode	enable ip_mode	reserved	reserved
Profile Velocity Mode	reserved	reserved	reserved
Servo Mode	enable servo_mode	reserved	reserved

1.2.2. STATUS WORD

Index	**0x6041**
Name	Status Word
Object Code	VAR
Data Type	Unsigned16
Object Class	all
Access	ro
PDO Mapping	Possible
Default Value	-

The status word indicates the current status of the drive. It is possible to define the TPDO to be transmitted at every change of status word (Device Event transmission type).

Bit Number	**Function**
0	Ready to Switch On
1	Switch On
2	Operation Enabled
3	Fault
4	Voltage Enabled
5	Quick Stop
6	Switch On Disabled
7	Warning
9	Remote
10	Target Reached
12	Operation Mode Specific
13	Operation Mode Specific
15	Manufacturer Specific: Drive Busy

Device Status Bit Meaning:

State	**Bit 6 Switch On Disable**	**Bit 5 Quick Stop**	**Bit 3 Fault**	**Bit 2 Operation Enable**	**Bit 1 Switched On**	**Bit 0 Ready to Switch On**
Not Ready to Switch On	0	X	0	0	0	0
Switch On Disabled	1	X	0	0	0	0
Ready to Switch On	0	1	0	0	0	1
Switched On	0	1	0	0	1	1
Operation Enable	0	1	0	1	1	1
Fault	0	X	1	1	1	1
Fault Reaction Active	0	X	1	1	1	1
Quick Stop Active	0	0	0	1	1	1

Bits 12, 13 are operation mode specific:

Mode	**Bit 12**	**Bit 13**	
Profile Position Mode	setpoint acknowledge	Following Error	
Homing Mode	Homing attained	Homing error	
Interpolated Position Mode	Ip-Mode active	reserved	
Profile Velocity Mode	Speed = 0	reserved	
Servo Mode	servo_mode active	reserved	

1.3. QUICK STOP OPTION CODES: *to be documented*

1.4. Mode of OPERATION: *control and display*

Index	0x6060
Name	Mode of Operation
Object Code	VAR
Data Type	integer8
Object Class	all
Access	rw
Save	Yes
PDO Mapping	Yes

This parameter changes the operation mode of the drive.

Mode of Operation	Action
1	Profile Position Mode (PP)
3	Profile Velocity Mode (PV)
4	Profile Torque Mode (PT)
6	Homing Mode (HM)
7	Interpolated Position Mode (IP)
8	Cyclic Synchronous Position Mode (CSP)
9	Cyclic Synchronous Velocity Mode (CSV)
10	Cyclic Synchronous Torque Mode (CST)
-4	Servo Mode (SM)

The actual mode is reflected in the operation mode display (object 6061h).

Index	0x6061
Name	Mode of Operation Display
Object Code	VAR
Data Type	integer8
Object Class	all
Access	ro
PDO Mapping	Yes
Default Value	7

1.5. ERROR CODES

Index	0x3022
Name	Error monitoring
Object Code	ARRAY
Number of Elements	3

Value Description

Sub Index	1
Description	Error monitoring
Data Type	Unsigned32
Object Class	all
Access	ro
PDO Mapping	No
Value	See below
Default value	No

Bit	Mask	Function
0	0x00000001	Hardware System 2 Error
1	0x00000002	24 Volt Error
2	0x00000004	Undervolt (time-delayed)
3	0x00000008	Braking system error
4	0x00000010	Safety channel 2 Error
5	0x00000020	Overvoltage
6	0x00000040	Internal Communication Error
7	0x00000080	Short-circuit
9	0x00000200	Mains interruption
10	0x00000400	Power Module overtemperature
12	0x00001000	Fan
16	0x00010000	Current measurement offset
17	0x00020000	Overcurrent
18	0x00040000	Encoder counting error
19	0x00080000	RDC Resolver
20	0x00100000	Resolver (cable interrupted)
21	0x00200000	Encoder (cable interrupted)
21	0x00400000	Encoder (Z marker pulse)
28	0x10000000	Manufacturer parameters error
29	0x20000000	Internal Communication error
30	0x40000000	Configuration error
31	0x80000000	System error

Sub Index	2
Description	Error monitoring
Data Type	Unsigned32
Object Class	all
Access	ro
PDO Mapping	No
Value	See below
Default value	No

Bit	Mask	Function
0	0x00000001	Torque following error
1	0x00000002	Speed following error
2	0x00000004	Position following error
4	0x00000010	Motor Temperature
5	0x00000020	I^2t
7	0x00000080	Busy
8	0x00000100	Calibration parameters error
9	0x00000200	Drive parameters error
10	0x00000400	User parameters error
16	0x00010000	SYNC cycle error
17	0x00020000	IP reference underflow/overflow
18	0x00040000	Bus error (Node guarding, Heartbeat...)
20	0x00100000	SD card error
21	0x00200000	File Erase/Write Error
22	0x00400000	Computation overflow
23	0x00800000	Safety channel 1 Error
31	0x80000000	Procedure error (autotuning, autophasing...)

1.6. WARNING CODES

Index	**0x3024**
Name	Warning Code
Object Code	VAR
Data Type	Unsigned32
Object Class	all
Access	ro
PDO Mapping	Possible
Default Value	0

Bit	**Mask**	**Function**
0	0x00000001	STO active
9	0x00000200	Main power off
10	0x00000400	IGBT module temperature
12	0x00001000	Fan failure
18	0x00040000	I^2t
21	0x00200000	Motor temperature

2. GENERAL

2.1. MANUFACTURER DRIVE DATA

Index	**0x6510**
Name	Manufacturer Drive Data
Object Code	ARRAY
Number of Elements	2

This object indicates the peak current and the rated current supported by the power module.

Value Description

Sub Index	1
Description	Drive Max. Current Defines the drive current rating
Data Type	Unsigned32
Access	ro
PDO Mapping	No
Unit	mA

Sub Index	2
Description	Drive Rated Current Defines the drive rated current
Data Type	Unsigned32
Access	ro
PDO Mapping	No
Unit	mA

Sub Index	3
Description	Drive Voltage Defines the drive rated voltage
Data Type	Unsigned16
Access	ro
PDO Mapping	No
Unit	V

Sub Index	4
Description	Drive Service Voltage Defines the drive operating voltage
Data Type	Unsigned16
Access	rw
PDO Mapping	No
Unit	V
Value	Must be less than or equal to Drive Voltage (0x6510-3)

2.2. MOTOR DATA

Index	**0x6410**
Name	Motor Data
Object Code	RECORD
Object Class	all
Number of Elements	20

This object defines the motor manufacturer' motor data.

Value Description

Sub Index	1
Description	Motor Manufacturer Name
Data Type	String
Access	rw
PDO Mapping	No
Value	Maximum 30 characters

Sub Index	2
Description	Motor Model Name
Data Type	String
Access	rw
PDO Mapping	No
Value	Maximum 30 characters

Sub Index	3
Description	Motor Code Special code or personalisation code.
Data Type	String
Access	rw
PDO Mapping	No
Value	Maximum 30 characters

Sub Index	4
Description	Catalog Date Code
Data Type	Unsigned16
Access	rw
Object Class	all
PDO Mapping	No

The structure of the entries is the following:

MSB LSB

Year (7-bit)	Month (4-bit)	Date (5-bit)

Year is relative to 1984.

Sub Index	5
Description	Modification Date Code
Data Type	Unsigned16
Access	rw
PDO Mapping	No

Sub Index	6
Description	Motor Type
Data Type	Unsigned16
Access	rw
PDO Mapping	No
Value	0 Rotative 1 Linear

Sub Index	7
Description	Motor Max Speed
Data Type	Unsigned32
Access	rw
PDO Mapping	No
Unit	rpm

Sub Index	8
Description	Motor Rated Speed
Data Type	Unsigned32
Access	rw
PDO Mapping	No
Unit	rpm

Sub Index	9
Description	Motor Stall Current
Data Type	Unsigned32
Access	rw
PDO Mapping	No
Unit	mA

Sub Index	10
Description	Motor Peak Current
Data Type	Unsigned32
Access	rw
PDO Mapping	No
Unit	mA

Sub Index	11
Description	Torque Constant (Kt)
Data Type	Unsigned16
Access	rw
PDO Mapping	No
Unit	0.001Nm/A

Sub Index	12
Description	Inertia
Data Type	Unsigned16
Access	rw
PDO Mapping	No
Unit	0.001gm²

Sub Index	13
Description	Inductance
Data Type	Unsigned16
Access	rw
PDO Mapping	No
Unit	0.1mH

Sub Index	14
Description	Number of motor pole pairs
Data Type	Unsigned16
Access	rw
PDO Mapping	No
Value	1..24

Sub Index	15
Description	Motor Phase
Data Type	Unsigned16
Access	rw
PDO Mapping	No
Value	0x5555 or 0xAAAA

Sub Index	16
Description	Motor Calage
Data Type	Unsigned16
Access	rw
PDO Mapping	No
Value	

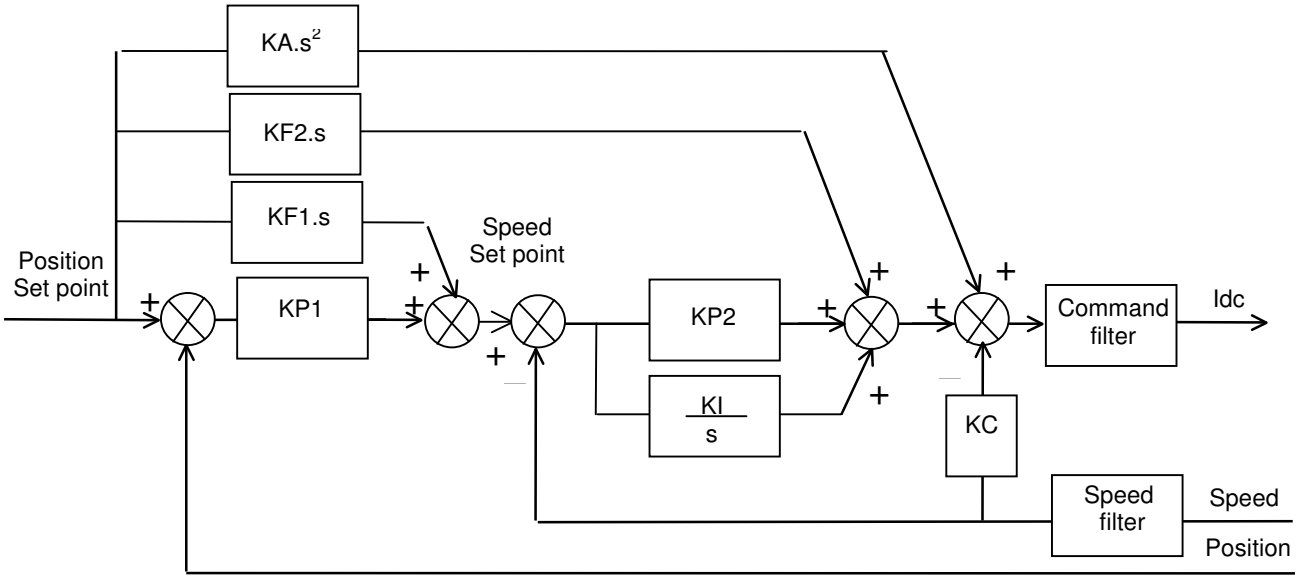
Sub Index	17
Description	Motor Temperature Probe
Data Type	Integer16
Access	rw
PDO Mapping	No
Value	1 PTC -1 NTC 0 not defined

Sub Index	18
Description	Motor Temperature Probe Threshold 1
Data Type	Unsigned16
Access	rw
PDO Mapping	No
Value	

Sub Index	19
Description	Motor Temperature Probe Threshold 8
Data Type	Unsigned16
Access	rw
PDO Mapping	No
Value	

2.3. SERVOLOOPS

2.3.1. REGULATOR



Index	Object	Name	Type	Attr.
0x60FB	RECORD	Position Loop Gain		
0x30FC	ARRAY	Position Loop Monitoring		
0x60B0	VAR	Position Offset		
0x6062	VAR	Position Demand Value		
0x60F4	VAR	Following Error Actual Value		
0x6063	VAR	Actual position*		
0x6064	VAR	Actual position		
0x6065	VAR	Following Error Window		

3. OPERATION MODES

3.1. PROFILE POSITION MODE (PP)

In this mode, a trapezoidal trajectory generator gives the drive the possibility to execute a positioning with preset parameters as target position, profile speed and acceleration.

3.1.1. OBJECT DICTIONARY ENTRIES

Index	Object	Name	Type	Attr.
0x607A	VAR	Target Position	Integer32	rw
0x6080	VAR	Max. Motor Speed	Unsigned16	rw
0x6081	VAR	Profile Velocity	Unsigned32	rw
0x6082	VAR	End Velocity	Unsigned32	rw
0x6083	VAR	Profile Acceleration	Unsigned32	rw
0x6084	VAR	Profile Deceleration	Unsigned32	rw
0x6086	VAR	Motion Profile Type	Integer16	rw
0x6067	VAR	Position Window	Unsigned32	rw
0x6068	VAR	Position Window Time	Unsigned16	rw
0x607F	VAR	Max. Profile Velocity	Unsigned32	rw

3.1.2. TARGET POSITION

Index	**0x607A**
Name	Target Position
Object Code	VAR
Data Type	Integer32
Object Class	pp
Access	rw
PDO Mapping	Yes
Unit	User Position Unit
Value Range	$(-2^{31})..(2^{31}-1)$
Default Value	0

Target position is the final position where the motor will move to in profile position mode. The start position is the current position. The positioning starts with a rising edge of bit 4 of the control word (new set point). Bit 6 of the control word indicates if the target position is an absolute (=0) or relative (=1) movement.

3.1.3. MAXIMUM MOTOR SPEED

Index	**0x6080**
Name	Max. Motor Speed
Object Code	VAR
Data Type	Integer32
Object Class	all
Access	rw
PDO Mapping	No
Unit	rpm
Value Range	100...60000
Default Value	3000

The *max. motor speed* defines the maximum speed which the drive can reach. To avoid a saturation of the servo loop, the running speed must be less than the *max. motor speed* (depends on the overshoot accepted for the servo loop response).
This parameter modifies the value of Max. Profile Velocity 0x607F.

3.1.4. PROFILE VELOCITY

The *Profile Velocity* is the running velocity for a positioning. If the positioning is too short, the profile velocity may not be reached.

Index	**0x6081**
Name	Profile Velocity
Object Code	VAR
Data Type	Unsigned32
Object Class	pp
Access	rw
PDO Mapping	Possible
Unit	User Velocity Unit
Value Range	-
Default Value	0x1000

3.1.5. END VELOCITY

The *End Velocity* is the final velocity value when the target position is reached. If the motor must stop at the target position, *End Velocity=0*.

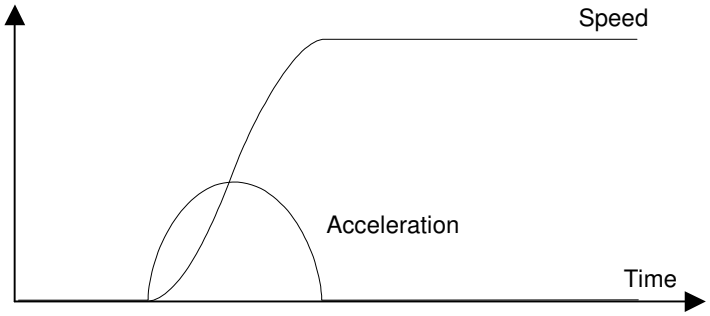
Index	**0x6082**
Name	End Velocity
Object Code	VAR
Data Type	Unsigned32
Object Class	pp
Access	rw
PDO Mapping	Possible
Unit	User Velocity Unit
Value Range	-
Default Value	0

3.1.6. PROFILE ACCELERATION/DECELERATION

Index	**0x6083**	**0x6084**
Name	Profile Acceleration	Pfile deceleration
Object Code	VAR	
Data Type	Unsigned32	
Object Class	pp	
Access	rw	
PDO Mapping	No	
Unit	Inc/s^2	
Value Range	-	
Default Value	0x10000	

3.1.7. MOTION PROFILE TYPE

Index	**0x6086**
Name	Motion Profile Type
Object Code	VAR
Data Type	Integer16
Object Class	pp
Access	rw
PDO Mapping	No
Value Range	0 -> Trapezoidal profile -1 -> S-Curve
Default Value	0



The S-curve is defined by a polynomial. The acceleration profile is therefore parabolic

3.1.8. POSITION WINDOW

The *Position Window* defines a symmetrical range of accepted positions relatively to the target position. If the motor current position is within the position window, this target position is considered as reached (bit 10 os status word - Target Reached – is set). If the position window value is 0, the position window control is not active.

Index	**0x6067**
Name	Postion Window
Object Code	VAR
Data Type	Unsigned32
Object Class	pp
Access	rw
PDO Mapping	No
Unit	User Position Unit
Default Value	0

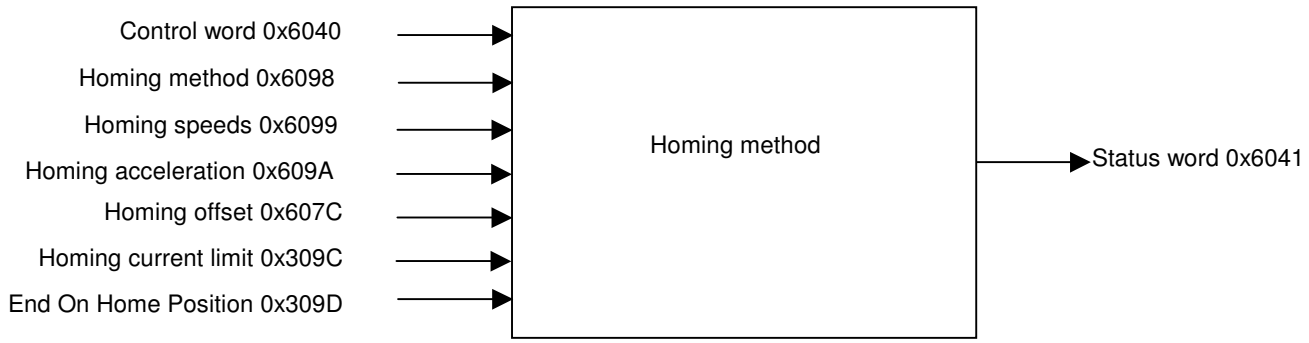
When the actual position is within the *Position Window* during the defined *Position Window Time*, the corresponding bit 10 *Target reached* in the *StatusWord* will be set at 1.

Index	**0x6068**
Name	Position Window Time
Object Code	VAR
Data Type	Unsigned16
Object Class	pp
Access	rw
PDO Mapping	Possible
Unit	Milliseconds
Value Range	0...32767
Default Value	0

3.2. HOMING MODE (HM)

3.2.1. INTRODUCTION

When the feedback sensor does not give the absolute position, the homing mode is the right way to set up the motor to a known position. This position can be detected using several signals such as positive or negative switch limit, home switch, index pulse or mechanical limit. The choice of the homing method depends on those signals and on the direction of the starting movement.
According to the homing method, the drive generates the trajectory. This is the reason why the position loop of the drive is employed.
Diagram of the trajectories as a function of the input signals:



Index	Object	Name	Type	Attr.
0x607C	VAR	Home Offset	Integer32	rw
0x6098	VAR	Homing Method	Integer8	rw
0x6099	ARRAY	Homing Speeds	Unsigned32	rw
0x609A	VAR	Homing Acceleration	Unsigned32	rw

Manufacturer Specific Objects:

Index	Object	Name	Type	Attr.
0x309C	VAR	Homing Current Limit	Unsigned16	rw
0x309D	VAR	End On Home Position	Unsigned16	rw

The homing procedure is launched on rising edge of bit 4 of the Control Word and can be interrupted when clear.
Meanings of operation mode specific bits of the Status Word:

Bit 13	Bit 12	Bit 10	Definition
0	0	0	Homing procedure is in progress
0	0	1	Homing procedure is interrupted or not started
0	1	0	Homing is reached, but target is not reached
0	1	1	Homing procedure is successfully completed
1	0	0	Homing error occured, velocity is not 0
1	0	1	Homing error occurred, velocity is 0
1	1	X	reserved

If Bit 10 is set, this indicates that the velocity is 0.
If bit 12 is set, this indicates that the home position is known but not available.
Bit 12 is reset at 0:
at power-up,
in case of a sensor fault,
on homing error,
when the homing is starting,
when the device control state is neither in "Switch On" state nor in "Operation Enabled" state.

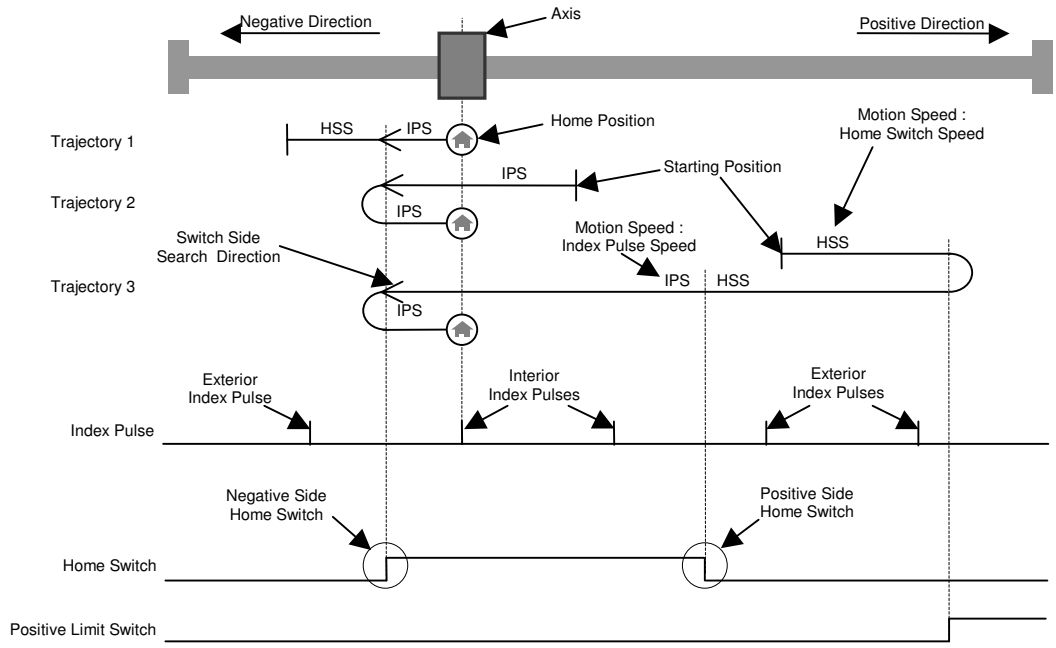
Bit 13 indicates a homing error:
homing launched whereas the drive is not in "Operation enabled" (except for homing method 35);
homing launched with an unimplemented selected method.
Bit 13 is reset at zero:
at the drive power up,
on rising edge of bit 7 of the Control Word.

3.2.2. HOMING METHOD

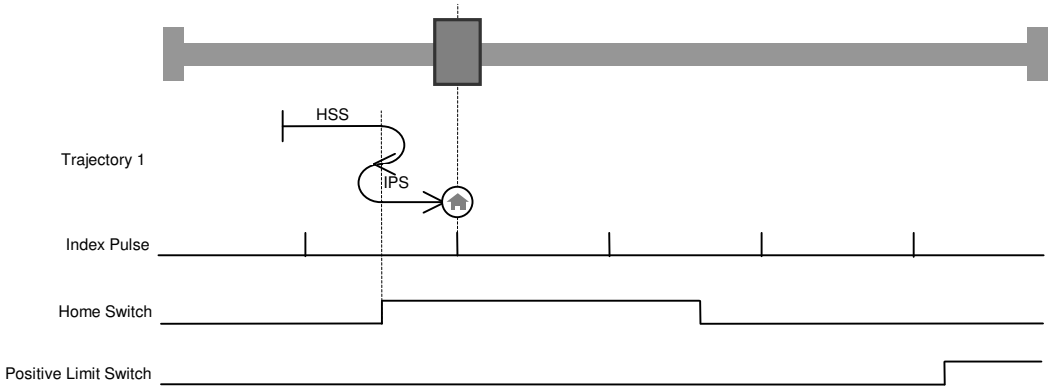
The *Homing Method* defines various ways of the drive to search the homing position.

Index	0x6098
Name	Homing Method
Object Code	VAR
Data Type	Integer8
Object Class	hm
Access	rw
PDO Mapping	No
Default Value	23h

Each homing method is detailed using a diagram representing all of the possible trajectories. Homing Method 8 is taken as an example for explaining the legend.



For simplifying diagrams, the trajectory of the switch side searching is not explicitely drawn. However, an arrow indicates the direction used to search a switch side. So, trajectory 1 of homing method 9 is explained in the following diagram:



The following explanation describes only the trajectory 1 of homing method 9 taken above as an example. When using homing method 9, the initial motion direction is positive, except if the home switch is active at the motion start. So, the negative side of the home switch is at first searched in the positive direction with the Home Switch Speed. When the activation of the home switch is detected, the drive reverses for searching the home switch deactivation. When the home switch has been found, the speed is the slowest home speed, namely the Index Pulse Speed. Once the deactivation of the home switch has been found, the drive reverses to position for searching the Index Pulse. At this stage, depending on the position sensor, the home position will directly be reached, e.g. a resolver. For sensors like incremental encoders, a search of Index Pulse is achieved in the positive direction and then the drive reverses to position on the captured Index Pulse position.

3.2.3. AVAILABLE METHODS

Supported standard methods: 1..14, 17..30, 33..35.
Specific methods: -1, -2, -3, -4.

Method	Search for Switch	Search for Index Pulse	Remarks
1	Negative Limit Switch	Exterior	
2	Positive Limit Switch	Exterior	
3	Home Switch, Negative Side	Exterior	
4	Home Switch, Negative Side	Interior	
5	Home Switch, Positive Side	Exterior	
6	Home Switch, Positive Side	Interior	
7	Home Switch, Negative Side	Exterior	Positive Initial Move. Reverse direction on Positive Limit Switch.
8	Home Switch, Negative Side	Interior	Positive Initial Move. Reverse direction on Positive Limit Switch.
9	Home Switch, Positive Side	Interior	Positive Initial Move. Reverse direction on Positive Limit Switch.
10	Home Switch, Positive Side	Exterior	Positive Initial Move. Reverse direction on Positive Limit Switch.
11	Home Switch, Positive Side	Exterior	Negative Initial Move. Reverse direction on Negative Limit Switch.
12	Home Switch, Positive Side	Interior	Negative Initial Move. Reverse direction on Negative Limit Switch.
13	Home Switch, Negative Side	Interior	Negative Initial Move. Reverse direction on Negative Limit Switch.
14	Home Switch, Negative Side	Exterior	Negative Initial Move. Reverse direction on Negative Limit Switch.
17	Negative Limit Switch	-	
18	Positive Limit Switch	-	
19	Positive Home Switch	-	
20	Positive Home Switch	-	
21	Negative Home Switch	-	
22	Negative Home Switch	-	
23	Home Switch, Negative Side	-	
24	Home Switch, Negative Side	-	
25	Home Switch, Positive Side	-	
26	Home Switch, Positive Side	-	
27	Home Switch, Positive Side	-	
28	Home Switch, Positive Side	-	
29	Home Switch, Negative Side	-	
30	Home Switch, Negative Side	-	
33		First Index Pulse	Negative Initial Move.
34		First Index Pulse	Positive Initial Move.
35		-	Homing On Current Position
-1	Mechanical Limit, Negative Move	First Index Pulse	
-2	Mechanical Limit, Positive Move	First Index Pulse	
-3	Mechanical Limit, Negative Move	-	
-4	Mechanical Limit, Positive Move	-	

3.2.4. HOMING PARAMETERS

- **HOMING OFFSET**

The *Home Offset* defines the position feedback value when the motor reaches the home position.

Index	**0x607C**
Name	Home Offset
Object Code	VAR
Data Type	Integer32
Object Class	hm
Access	rw
PDO Mapping	No
Unit	Inc (Revolution increments: object 608F sub-index 1)
Value Range	$(-2^{31})..(2^{31}-1)$
Default Value	0

- **HOMING SPEEDS**

***Homing Speeds* defines the motor speed during the various searching steps of the home position.**

Index	**0x6099**
Name	Homing Speeds
Object Code	ARRAY
Number of Elements	2
Data Type	Unsigned32

Value Description

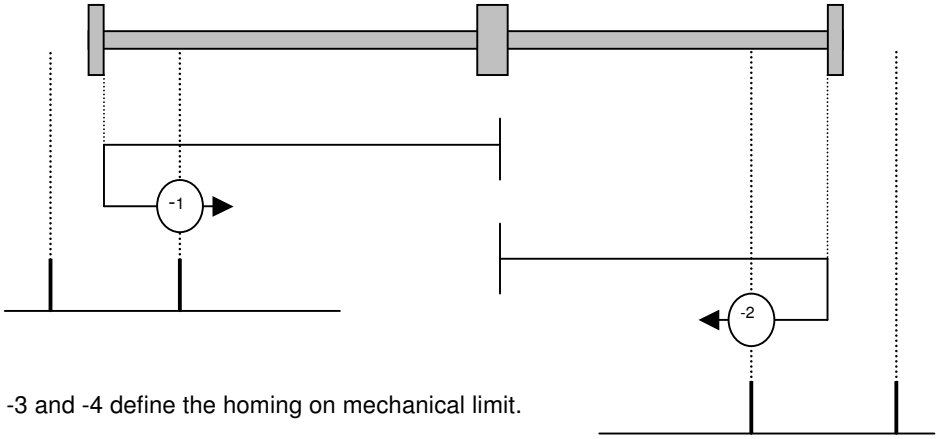
Sub Index	1
Description	Speed during search of switch
Object Class	hm
Access	rw
PDO Mapping	No
Unit	inc/s
Default Value	00000019h

Sub Index	2
Description	Speed during search of zero
Object Class	hm
Access	rw
PDO Mapping	No
Unit	inc/s
Default Value	0000000Ah

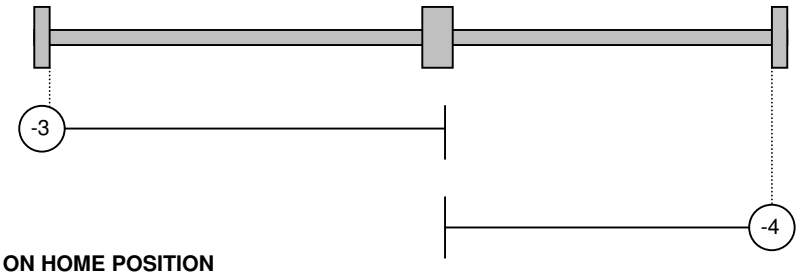
- **HOMING CURRENT LIMIT**

Index	**0x309C**
Name	Homing Current Limit
Object Code	VAR
Data Type	Unsigned16
Object Class	hm
Access	rw
PDO Mapping	No
Unit	%
Conversion	0 to 3FFFh -> 0% to 100 %
Default Value	400h

The "Homing current limit" defines the limit of current during homing on the mechanical limit. The value is defined as a percent of the drive maximum current (defined by object 6510h sub-index 5).
FUNCTIONAL DESCRIPTION
The "Homing Current Limit" parameter defines the motor current limit during the homing procedure. When the mechanical limit is reached, the current in the motor raises up to this limit and the motor speed is 0. This position will be taken as the homing position. An offset value (object 607Ch) can be used to preset the homing position value.
Method -1 and -2 define the homing on mechanical limit with index pulse.



Method -3 and -4 define the homing on mechanical limit.



- **END ON HOME POSITION**

This parameter allows the drive not to reverse when the homing is over.

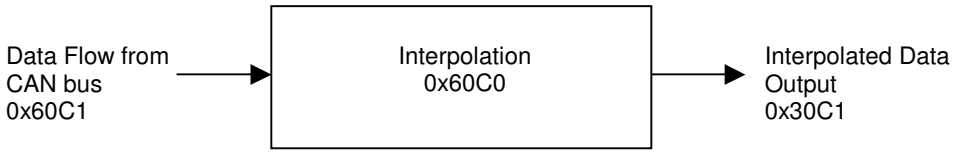
If it is set at 1, make a move to the home position when the homing is finished. If clear, the home position is found but not moved to.

Index	**309Dh**
Name	End on Home Position
Object Code	VAR
Data Type	Unsigned16
Object Class	hm
Access	rw
PDO Mapping	No
Default Value	1

3.3. INTERPOLATED POSITION MODE (IP)

3.3.1. DEFINITION

The interpolated position mode is used to control several axes in coordination. The trajectory must be generated by the host controller and the elementary set point is sent at a fixed cycle time (same as communication cycle time) to all axes.
The cycle time synchronisation of all axes is ensured by SYNC message. The set point data flow must be sent in real-time.
The elementary set point could be only position if linear interpolation is chosen. The PV interpolation mode requires position and velocity for each set point.



In interpolated position mode, the interpolated data output will be applied to the position loop.

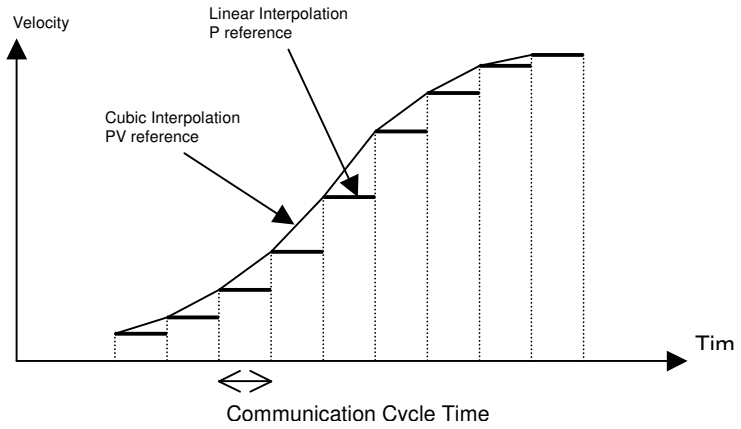
Index	Object	Name	Type	Attr.
0x60C0	VAR	Interpolation Submode Select	Integer16	rw
0x60C1	RECORD	Interpolation Data Record		rw
0x60C4	RECORD	Interpolation Data Configuration		rw
0x30C1	VAR	Interpolated Data Output	Integer32	rw

3.3.2. INTERPOLATION SUBMODE SELECT

Index	0x60C0
Name	Interpolation Submode Select
Object Code	VAR
Data Type	Integer16
Object Class	ip
Access	rw
PDO Mapping	No
Default Value	0

Interpolation Submode Select	Description
0	Linear interpolation
-1	PV interpolation
-2	P3 interpolation

In linear interpolation mode, only the first parameter of interpolation data record is used. The data must be the position reference.
In PV interpolation mode, the first parameter of interpolation data record must contain the position reference and the second parameter of interpolation data record contains the velocity reference.



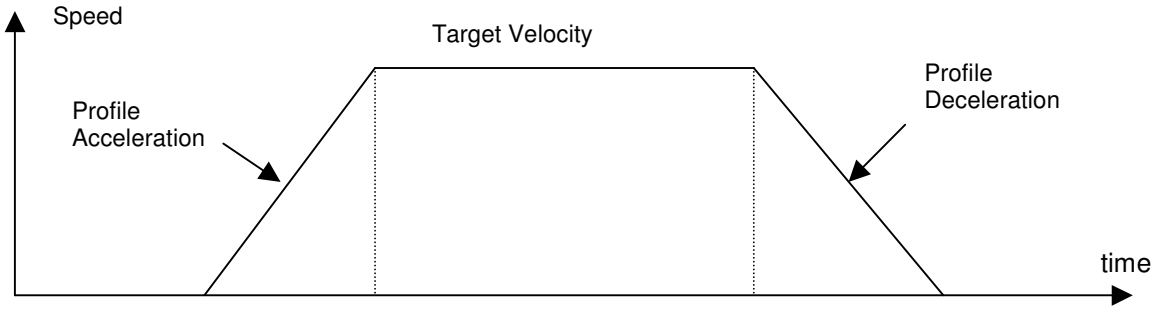
Note: The velocity reference for each set-point must be the instantaneous velocity at this point (and not the average velocity).

3.3.3. DATA RECORDS TO BE DOCUMENTED

3.4. PROFILE VELOCITY MODE (PT)

3.4.1. DEFINITION

The profile velocity mode authorizes the drive to operate with a velocity reference. Only speed loop and current loop are closed in this mode.



Index	Object	Name	Type	Attr.
0x606B	VAR	Velocity Demand Value	Integer32	ro
0x606C	VAR	Velocity Actual Value	Integer32	ro
0x60FF	VAR	Target Velocity	Integer32	rw
0x6083	VAR	Profile Acceleration	Unsigned32	rw
0x6084	VAR	Profile Deceleration	Unsigned32	rw
0x606D	VAR	Velocity Window	Unsigned16	rw
0x606E	VAR	Velocity Window Time	Unsigned16	rw
0x606F	VAR	Velocity Threshold	Unsigned16	rw
0x6070	VAR	Velocity Threshold Time	Unsigned16	rw
0x30FF	VAR	Target Velocity Source	Unsigned16	rw

The *Velocity Window* defines a symmetrical range of accepted velocity relatively to the target velocity. If the motor current velocity is within the velocity window, this target velocity is considered as reached (bit 10 of the status word - Target Reached – is set). If the velocity window value is 0, the velocity window control is not active.

When the actual velocity is within the *Velocity Window*

during the defined *Velocity Window Time*, the corresponding bit 10 *Target reached* in the *StatusWord* will be set at 1.

Index	**0x606D**
Name	Velocity Window
Object Code	VAR
Data Type	Unsigned32
Object Class	pv
Access	rw
PDO Mapping	No
Unit	Velocity Unit
Default Value	0

Index	**0x606E**
Name	Velocity Window Time
Object Code	VAR
Data Type	Unsigned16
Object Class	pv
Access	rw
PDO Mapping	Possible
Unit	ms
Value Range	0...32767
Default Value	0

The *Velocity Threshold* defines a symmetrical range of accepted velocity relatively to 0. If the motor current velocity is within the velocity threshold, this 0 velocity is considered as reached (bit 12 of status word - Velocity = 0 – is set). If the velociy threshold value is 0, the velocity threshold control is not active.

Index	**0x606F**
Name	Velocity Threshold
Object Code	VAR
Data Type	Unsigned32
Object Class	pv
Access	rw
PDO Mapping	No
Unit	Velocity Unit
Default Value	0

When the actual velocity is within the *Velocity Threshold* during the defined *Velocity Threshold Time*, the corresponding bit 12 *Velocity=0* in the *StatusWord* will be set at 1.

3.4.2. PROFILE VELOCITY MODE INPUT SOURCE

This object allows to connect any 32-bit dataflow as target velocity for the Profile Velocity Mode. The structure of the entries is the following**:**

Index	**0x30FF**
Name	Profile Velocity Mode Input Source for Target Velocity
Description	Index/sub-index of input data
Data Type	Unsigned32
Class	Pv
Access	Rw
PDO Mapping	No
Value	See below
Default Value	0x60FF0000

MSB		LSB
Index (16-bit)	Sub-index (8-bit)	0

Example:

0x30FF,0 = 0x30F10200

Connects the analog input as target velocity for Profile Velocity Mode.

3.5. PROFILE TORQUE MODE (PT)

In this mode, the drive operates only with current loops and there is no speed or position control**.**

Object Dictionary Entries

Index	**0x6070**
Name	Velocity Threshold Time
Object Code	VAR
Data Type	Unsigned16
Object Class	pv
Access	rw
PDO Mapping	Possible
Unit	ms
Value Range	0...32767
Default Value	0

Index	**Object**	**Name**	**Type**	**Attr.**
0x6071	VAR	Target Torque	Integer16	rw
0x3071	VAR	Target Torque Input Source	Unsigned32	rw
0x6087	VAR	Torque Slope	Unsigned32	rw
0x6088	VAR	Torque Profile Type	Integer16	rw
0x60B2	VAR	Offset Torque	Integer16	rw
0x6074	VAR	Torque Demand Value	Integer16	ro
0x6077	VAR	Torque Actual Value	Integer16	ro
0x6078	VAR	Current Actual Value	Integer16	ro
0x6079	VAR	DC Voltage	Integer16	ro

3.5.1. PROFILE TORQUE MODE INPUT SOURCE

Index	0x3071
Name	Profile Torque Mode Input Source for Target Torque
Description	Index/sub-index of input data
Data Type	Unsigned32
Class	Pt
Access	Rw
PDO Mapping	No
Value	See below
Default Value	0x60710000

This object allows to connect any 16-bit dataflow as a target torque for the Profile Torque Mode.
The structure of the entries is as follows:

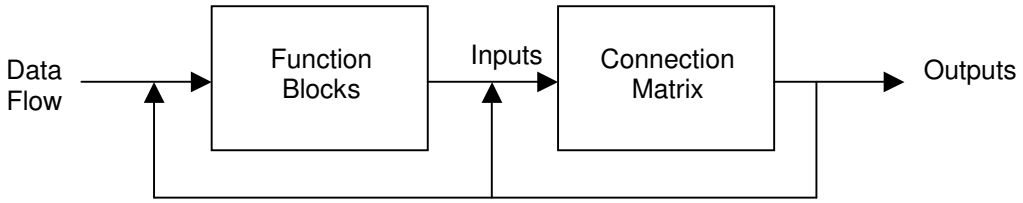
MSB		LSB
Index (16-bit)	Sub-index (8-bit)	0

Example:
0x3071,0 = 0x30F10100
Connects analog input 1 as a target torque for the Profile Torque Mode

3.6. SERVO MODE (SM)

3.6.1. DEFINITION

The Servo mode is a manufacturer specific mode which allows to easily combine various motion functions in order to solve complex motion applications.
The servo mode function block is connected as below:



- **THE CONNECTION MATRIX IS DEFINED BY:**

- 8 inputs
- 4 outputs
- Connections between inputs and outputs

- **INPUTS**

Each input is defined either by:

- a variable (index, sub-index). This variable can therefore come from any dynamic dataflow (fieldbus, data acquisition...) or from any function block (see below).
- an output.

The equation is INn(t) = OUTi(t-1)
This allows to connect output i to input n and to realize a delay function.
Each input and output is accessible by an object.
All 8 inputs can be connected to any mappable object by means of object 0x3501.

- **OUTPUTS**

There are 4 outputs. One of these outputs may be connected to the position setpoint (entry point of the position loop).
The definition of all outputs composes the connection matrix.
Each output is defined by an equation type:
OUTn(t) = INi(t) + INj(t) + INk(t) - INm(t)
i, j, k, m are any of the 8 inputs.

3.6.2. FUNCTION BLOCKS

Function Blocks are basic motion functions designed to realize simple functions:

Each fonction has:
- one or several inputs (defined by object index and sub-index)
- one output (defined by an object index and sub-index)
- function parameters.

Servo Mode objects

0x6040	Control Word
0x6041	Status Word
0x6060	Mode of Operation
0x6061	Mode of Operation Display

0x3500	SM Function Select
0x3501	SM Input connection configuration
0x3502	SM input signals
0x3503	SM outputs configuration
0x3504	SM output signals
0x3505	SM Position Setpoint Select
0x3506	SM Output Setpoint Value
0x3507	SM Control Word
0x3508	SM status Word
0x3509	*SM Pos Offset*

- **SERVO FUNCTION BLOCK SELECT**

This object allows to activate a funtion block.
Up to 4 function can be used at the same time.

Index	**0x3500**
Name	Servo Function Block Select
Object Code	ARRAY
Number of Elements	4

Value Description

Sub Index	1-4
Description	Function Select
Data Type	Unsigned8
Access	rw
PDO Mapping	No
Value	Function Number
Default value	0

Function Number:
0 Not used
1 Profile Generator
2 Profile Speed
3 Master Slave
4 Cam

These function blocks will be in the order defined by sub 1 to sub 4.
Example:
Application with Profile Speed and Cam
object 0x3501,1 = 2
object 0x3501,2 = 4
object 0x3501,3 = 0
object 0x3501,4 = 0
Profile Speed function will be executed first, then Cam function.

- **SM INPUTS**

Inputs connection configuration

This object allows to connect any dataflow to inputs of the connection matrix.
The connecting object must be a variable object (i.e. mappable object).
Value Description

Index	**0x3501**
Name	Inputs connection configuration
Object Code	ARRAY
Number of Elements	8

Sub Index	1-8
Description	Index/sub-index of input object data
Data Type	Unsigned32
Access	rw
PDO Mapping	No
Value	See below
Default value	0

The structure of the entries is the following:

MSB		LSB
Index (16-bit)	Sub-index (8-bit)	0

Example:
Input 1 is connected to interpolated data from CAN bus:
0x3501,1 = 0x30C10000
The interpolated data value from CAN bus is given in object 0x30C1,0

Index	**0x3502**
Name	Connection Input signals
Object Code	ARRAY
Number of Elements	8

- **SM INPUT SIGNALS**

This object allows to read the actual value of the input signal**.**

Value Description

Sub Index	1-8
Description	Input signal actual value
Data Type	Integer32
Access	ro
PDO Mapping	No
Value	-
Default value	-

- **SM OUTPUTS .**

Outputs configuration

Index	**0x3503**
Name	Outputs Configuration
Object Code	ARRAY
Number of Elements	4

This object allows to setup output signals.
Value Description

Sub Index	1-4
Description	Output 1-4 configuration
Data Type	Unsigned16
Access	rw
PDO Mapping	No
Value	-
Default value	-

Each output can be defined as a summary of maximum 4 inputs. The equation for output n is the **following:**

OUTn = INi + INj + INk - INm

MSB			LSB
m	k	j	i

Examples:
OUT3 = IN2 - IN3

Configuration Value of object 0x3503 sub-index 3 = 0x3002

- **SERVO MODE: OUTPUT SIGNALS**

This object allows to read the actual output value of the connection matrix.

Index	**0x3504**
Name	Output signals
Object Code	ARRAY
Number of Elements	8

Value Description

Sub Index	1-8
Description	Output signal actual value
Data Type	Integer32
Access	ro
PDO Mapping	No
Value	-
Default value	-

- **SM POSITION SETPOINT SELECTION**

This object allows to connect one of any outputs to the position loop.
The value is the output number of any outputs of the connection matrix.

Index	**0x3505**
Description	Position Setpoint Select
Data Type	Unsigned8
Access	rw
PDO Mapping	No
Value	1 .. 4
Default value	0

- **SERVO MODE: OUTPUT SETPOINT VALUE**

This object reads the value of the output which is connected to the position loop.

Index	**0x3506**
Description	Output setpoint value
Data Type	Integer32
Access	ro
PDO Mapping	Yes
Value	-
Default value	-

- **SM CONTROL & STATUS WORD**

Index	**0x3507**
Description	Control word for Servo Mode
Data Type	Unsigned32
Access	rw
PDO Mapping	Possible
Value	-
Default value	0

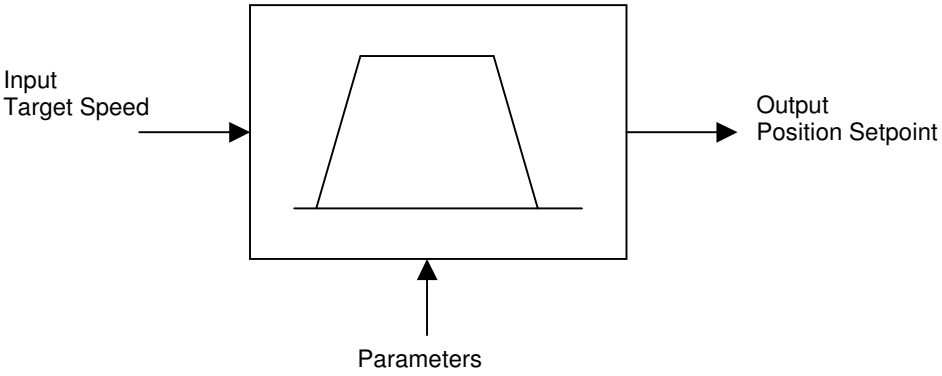
Bit Number	**Function**
0	Enable SetPoint This bit allows to connect the function output to the position loop setpoint
4	Start Profile Position
5	0 relative move 1 absolute move
8	Start Profile Speed
16	Start Cam

Index	0x3508
Description	Status word for Servo Mode
Data Type	Unsigned32
Access	ro
PDO Mapping	Possible
Value	-
Default value	0

- **SM POSITION OFFSET: TO BE DOCUMENTED**

3.7. Profile Speed Function Block

The Profile Speed Function generates the position setpoint with a user defined profile speed.



Input

- The Target speed can come from any dynamic object (master position from external encoder, fieldbus or internal generator...)

Parameters

- Acceleration: object 0x6083
- Deceleration: object 0x6084

Outputs

- Position Setpoint: this output can be connected to one of the inputs of the connection matrix.

Objects

0x3520	SM Profile Speed input source (target speed)
0x3526	SM Profile Speed output

3.7.1. PROFILE SPEED INPUT SOURCE (TARGET SPEED)

This object allows to connect any dataflow as target speed of the Profile Speed Function Block.
The structure of the entries is the following:

Index	0x3520
Name	Profile Speed Input Source for Target Speed
Description	Index/sub-index of input data
Data Type	Unsigned32
Object Class	sm
Access	rw
PDO Mapping	No
Value	See below

Bit Number	Function
0	Setpoint enabled
4	Profile Position setpoint acknownledge
5	Profile Position running
8	Profile Speed running
16	Cam Profile running

MSB LSB

Index (16-bit)	Sub-index (8-bit)	0

Example:
Input is connected to interpolated data from CAN bus:
0x3540,0 = 0x30C10000
The interpolated data value from CAN bus is given in object 0x30C1,0

- **PROFILE SPEED OUTPUT**

This object reads the actual value of the Profile Speed Function Block output.

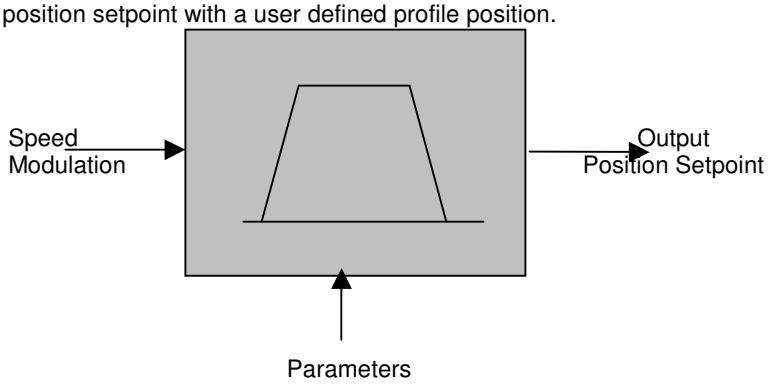
Index	**0x3526**
Description	Output of the Profile Speed Function Block
Data Type	Integer32
Access	ro
PDO Mapping	Yes
Value	-
Unit	Position Unit

3.8. Profile Generator Function Block

The Profile Generator Function generates the position setpoint with a user defined profile position.

Input
- Speed Modulation

Parameters
- The Target position
- Profile Acceleration
- Profile Deceleration
- Profile Velocity



0x3510	SM Profile Generator modulation input source
0x3514	SM Profile Generator parameter
0x3516	SM Profile Generator output

3.8.1. PROFILE GENERATOR SPEED MODULATION INPUT SOURCE

This object allows to connect any dataflow as as speed modulation of the Profile Generator Function Block.
The structure of the entries is the following:

Index	**0x3510**
Name	Profile Generator Speed Modulation Input Source
Description	Index/sub-index of input data
Data Type	Unsigned32
Object Class	sm
Access	rw
PDO Mapping	No
Default Value	0
Value	See below

MSB		LSB
Index (16-bit)	Sub-index (8-bit)	0

Profile Position Generator Parameters

Index	**0x3514**
Name	Profile Generator Parameters
Object Code	RECORD
Number of Elements	4

Value Description

Sub Index	1
Description	Profile Generator Target Position
Data Type	Integer32
Access	rw
PDO Mapping	Yes
Unit	Position Unit

Sub Index	2
Description	Profile Generator Velocity
Data Type	Unsigned32
Access	rw
PDO Mapping	No
Unit	Velocity Unit

Sub Index	4
Description	Profile Generator Deceleration
Data Type	Integer32
Access	rw
PDO Mapping	No
Unit	Acceleration Unit

3.9. Cam Function Block

The cam function allows to transform an input data flow into an output data flow defined by a table of values.

Input

- The main input defines the input data flow (master position from external encoder, fieldbus or internal generator...)

Parameters

- One shot cam (starts immediately)
- Continous cam. Cam table restarts when end is reached
- One shot cam triggered by an event (including time compensation)
- Continous cam triggered by an event (including time compensation)
- One shot cam triggered at a predefined input value
- Continous cam triggered at a predefined input value
- Absolute / relative cam
- Cam profile switching in real-time: switching between several cam tables.
- Input and output scaling possibilities.
- Output scaling by analog signal
- Linear/Cubic interpolation between table points
- Equidistant table (1 entry = output) or non equidistant table (2 entries = input and output)
- Cam speed defines the speed of cam table execution
- Bi-directional auto-repeat camming.

Outputs

- Cam Output
- Cam Status

Index	Sub-index	Name	Type	Attr.	
0x3540		Cam input source	Unsigned32	rw	Object index and sub-index

Index	Sub-index	Name	Type	Attr.	
0x3544		Cam Parameter	RECORD		
	0	Number of supported entries	Unsigned8	ro	7

Sub Index	3
Description	Profile Generator Acceleration
Data Type	Unsigned32
Access	rw
PDO Mapping	No
Unit	Acceleration Unit

	1	Cam Control	Unsigned16	rw	0: Enable/Disable 1: Start immediately 2: Start on Input value 3: Start on fast-input 4: Auto-repeat 5: Interpolation linear/cubic 7: Output factor select 8: Cam table select
	2	Cam Status	Unsigned16	ro	0: Run/Stopped
	3	Cam Table Select	Unsigned16	rw	
	4	Cam Input Factor	Unsigned32	rw	
	5	Cam Output Factor	Unsigned32	rw	
	6	Cam Input start	Integer32	rw	

Index	**Sub-index**	**Name**	**Type**	**Attr.**	
0x3546		Cam Output	Integer32	ro	

- **CAM INPUT SOURCE**

This object allows to connect any 32-bit dataflow as input of the cam function block.
The structure of the entries is the following:

Index	**0x3540**
Name	Cam Input Source
Description	Index/sub-index of input data
Data Type	Unsigned32
Access	rw
PDO Mapping	No
Value	See below

MSB		LSB
Index (16-bit)	Sub-index (8-bit)	0

Example:
Input is connected to interpolated data from CAN bus:
0x3540 = 0x30C10000
The interpolated data value from CAN bus is given in object 0x30C1,0

Cam Output

Index	**0x3546**
Description	Output of the Cam Function
Data Type	Integer32
Object Class	sm
Access	ro
PDO Mapping	Yes
Value	-
Unit	Position Unit

This object read the actual value of the Cam Function Block output.

3.10. SM GEARBOX Function block: to be documented

4. APPLICATION FEATURES

4.1. DIGITAL INPUTS & OUTPUTS

Digital Inputs / Outputs
Most of the digital IO's can be defined by parameters and access via standard CANopen objects. In the following description, take care not to make a confusion between "digital IO function" and physical layout.

Index	Object	Name	Type	Attr.
0x60FD	VAR	Digital Inputs	Unsigned32	ro
0x3050	ARRAY	Digital Inputs Configuration	Unsigned32	rw
0x3051	VAR	Digital Inputs Polarity	Unsigned32	rw
0x60FE	ARRAY	Digital Outputs	Unsigned32	rw
0x3054	ARRAY	Digital Outputs Configuration	Unsigned32	rw
0x3055	VAR	Digital Outputs Polarity	Unsigned32	rw

Digital Inputs value

Index	0x60FD
Name	Digital Inputs
Object Code	VAR
Data Type	Unsigned32
Object Class	all
Access	rw
PDO Mapping	Possible
Default Value	No

bit	Function
0	Logical Input Negative Limit Switch (not implemented)
1	Logical Input Positive Limit Switch (not implemented)
2	Logical Input HOME
3	Logical Input ENABLE
14	
15	
16	In1
17	In2
18	In3
19	In4 (fast input)
20	In5 (fast input)
21	In6
22	In7
23	In8
24	In9
25	
26	
27	
28	
29	
30	
31	

Digital Inputs Configuration

Index	0x3050
Name	Digital Inputs Configuration
Object Code	ARRAY
Number of Elements	9

The digital Inputs configuration allows to affect any digital input to one bit in a variable indicated by index and sub-index.

Value Description

Sub Index	i =1-8
Description	Digital Inputs Destination defines the destination object for the corresponding digital input.
Data Type	Unsigned32
Access	rw
PDO Mapping	No
Value Range	
Default Value	

- **THE STRUCTURE OF THE ENTRIES IS THE FOLLOWING:**

MSB		LSB
Index (16-bit)	Sub-index (8-bit)	Bit number n (0-15)

The state of the physical input i will be copied into bit n of object indicated by index and sub-index.
Digital Inputs Polarity

Index	**0x3051**
Name	Digital Inputs Polarity
Object Code	VAR
Data Type	Unsigned16
Object Class	all
Access	rw
PDO Mapping	Possible
Default Value	No

Digital Outputs valuesIndex	**0x60FE**
Name	Digital Output
Object Code	ARRAY
Number of Elements	2

Value Description

Sub Index	1
Description	Digital Output
Data Type	Unsigned32
Access	rw
PDO Mapping	Possible
Default Value	0

Digital Outputs Configuration

Index	**0x3054**
Name	Digital Outputs Configuration
Object Code	ARRAY
Number of Elements	4

The digital outputs configuration allows to affect one bit of any variable indicated by index and sub-index to a physical output.

Value Description

bit	**Function**	**Function**
0	In1	
1	In2	
2	In3	
3	In4 (fast input)	
4	In5 (fast input)	
5	In6	
6	In7	
7	In8	
8	In9	
9		
10		
11		
12		
13		
14		
15		

bit	
0	Motor Brake
1	
2	
3	
14	
15	
16	Physical Output 1
17	Physical Output 2
18	Physical Output 3
19	Physical Output 4
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	

Sub Index	2
Description	Digital Output Bitmask
Data Type	Unsigned32
Access	rw
PDO Mapping	No
Default Value	0

Sub Index	1-4
Description	Digital Output Source defines the source for digital output.
Data Type	Unsigned32
Access	rw
PDO Mapping	No
Value Range	
Default Value	

The structure of the entries is the following:

MSB LSB

Index (16-bit)	Sub-index (8-bit)	Bit number n (0-15)

The state of bit n of object index and sub-index will be copied to the physical output.

Digital Outputs Polarity

bit	**Function**
0	Physical Output 1
1	Physical Output 2
2	Physical Output 3
3	Physical Output 4
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	

Index	**0x3051**
Name	Digital Outputs Polarity
Object Code	VAR
ata Type	Unsigned16
Object Class	all
Access	rw
PDO Mapping	Possible
Default Value	No

4.2. ANALOG INPUTS

Index	**0x30F1, 0x30F2**
Name	Analog Input
Object Code	RECORD
Number of Elements	

Value Description

Sub Index	1
Description	Analog Input 16-bit Value Conversion data from ADC. The sampling rate is 16 kHz The result is left aligned.
Data Type	Integer16
Access	ro
PDO Mapping	Yes
Value Range	No
Default Value	No

Sub Index	2
Description	Analog Input 32-bit Value
Data Type	Integer32
Access	ro
PDO Mapping	Yes
Value Range	No
Default Value	No

Analog_Input_32bit_Value = (Analog_Input_16bit_Value - Offset) * Gain / 256

The Gain value is signed.
Example: using analog input as speed reference.
The speed reference is 32-bit, so the 32-bit value will be used.
The maximum speed is 30000 rpm and the unit is inc/s with 4096 inc per motor revolution.

Maximum speed: 30000 rpm -> 500 rev/s -> 2048000 inc/s
The maximum 16-bit analog input is 32767
Gain = 2048000 / 32767 * 256 = 16000

Sub Index	3
Description	Offset
Data Type	Integer16
Access	rw
PDO Mapping	Yes
Value Range	-
Default Value	0

Sub Index	4
Description	Gain
Data Type	Integer16
Access	rw
PDO Mapping	Yes
Value Range	-
Default Value	256

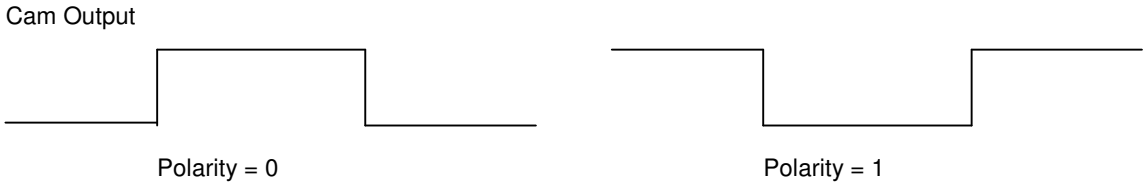
Sub Index	5
Description	Filter
Data Type	Unsigned16
Access	rw
PDO Mapping	Yes
Unit	Hz
Value Range	5-20000
Default Value	100

The filter is applied on Analog Input 16-bit Value.

4.3. DIGITAL CAM

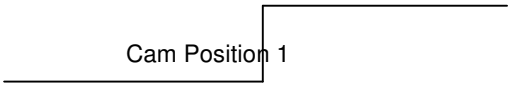
Index	Object	Name	Type	Attr.
0x30E0	ARRAY	Digital Cam configuration register	Unsigned16	rw
0x30E1	ARRAY	Digital Cam positions	Integer32	rw

Cams are fully defined with object 0x30E0 and 0x30E1. No parameters can be changed if Cam Enable Register is not 0.
Cam Polarity
Each bit of the Cam Polarity Register allows to set the polarity of the cam output. Normal polarity (polarity bit = 0) set the cam output with value 1 when the cam is active.



Cam Type

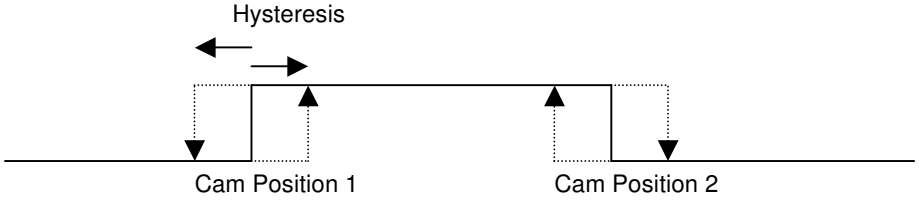
Each bit of the Cam Type Register defines the cam type.
CAM TYPE = 0: CAM DEFINED BY 1 POSITION.



- **CAM TYPE = 1: CAM DEFINED BY 2 POSITIONS.**

Cam Hysteresis

- **CAM HYSTERESIS REGISTER DEFINES A HYSTERESIS OF THE CAM POSITION.**



Index	**0x30E0**
Name	Digital Cam Positions
Object Code	ARRAY
Number of Elements	32

Digital Cam Positions can only be changed when Cam Enable Register = 0 (0x30E1-5).
Value Description

Sub Index	1	2
Description	First Position of Cam number 1	Second Position of Cam number 1
Data Type	Integer32	Integer32
Access	rw	rw
PDO Mapping	Yes	Yes
Value Range	No	No
Default Value	No	No

Digital Cam Configuration Registers

Index	**0x30E1**
Name	Digital Cam Configuration Registers
Object Code	ARRAY
Number of Elements	**32**

Registers with sub index 2 to 4 can only be changed when Cam Enable Register = 0.

Value Description

Sub Index	1
Description	Cam Status
Data Type	Unsigned16
Access	ro
PDO Mapping	Yes
Value Range	No
Default Value	No

Each bit of Cam status register corresponds to a Digital Cam (max. 16 cams)

Sub Index	2
Description	Cam Type
Data Type	Unsigned16
Access	rw
PDO Mapping	Yes
Value Range	No
Default Value	0

Each bit of Cam Type register corresponds to a Digital Cam (max. 16 cams)

0 Cam with 1 position

1 Cam with 2 positions

Sub Index	3
Description	Cam Polarity
Data Type	Unsigned16
Access	rw
PDO Mapping	Yes
Value Range	No
Default Value	0

Each bit of Cam Polarity register corresponds to a Digital Cam (max. 16 cams)

0 Cam with normal polarity

1 Cam with reversed polarity

Sub Index	4
Description	Cam Hysteresis
Data Type	Unsigned16
Access	rw
PDO Mapping	Yes
Value Range	No
Unit	position unit
Default Value	0

Sub Index	5
Description	Cam Enable
Data Type	Unsigned16
Access	rw
PDO Mapping	Yes
Value Range	No
Default Value	0

Each bit of Cam Polarity register corresponds to a Digital Cam (max. 16 cams)

0 Disable Cam

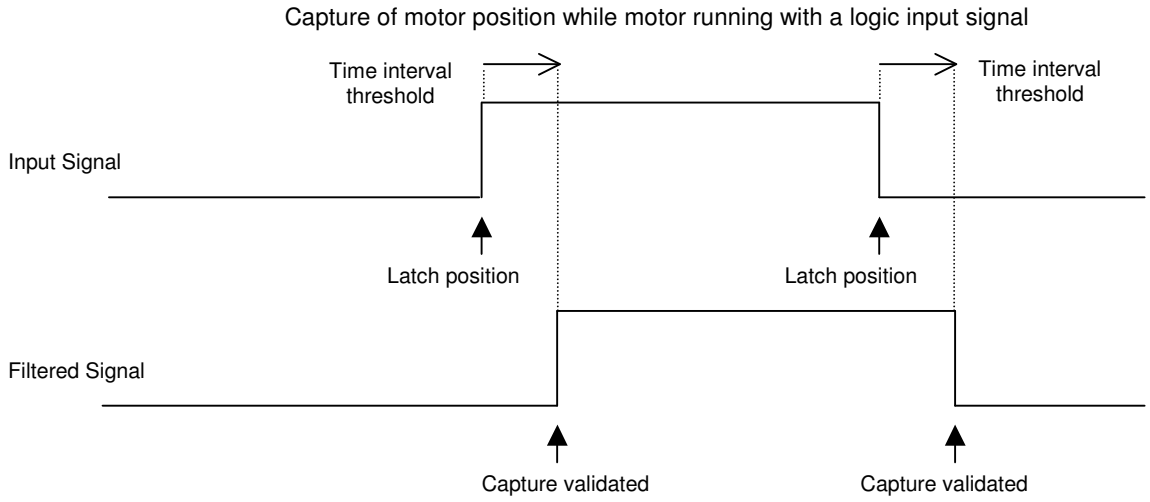
1 Enable Cam

4.4. POSITION CAPTURE

Index	Object	Name	Type	Attr.
0x3371	RECORD	Capture 1		
0x3372	RECORD	Capture 2		

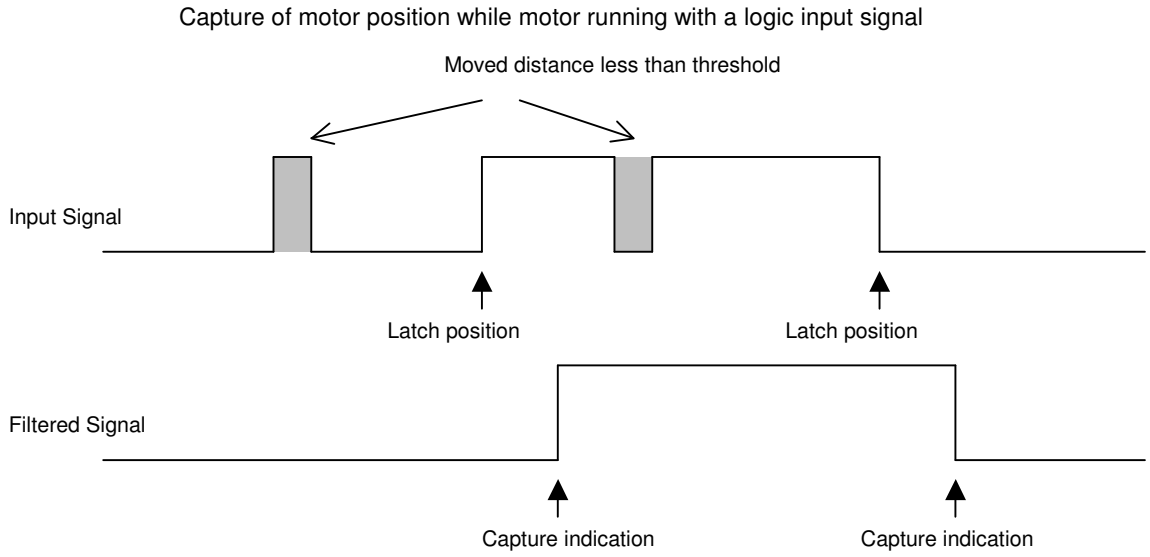
- **CAPTURE TIME FILTER**

This parameter defines the time interval threshold of the capture time filter. After the rising or the falling edge of the input signal, the input signal level must be stable for a time interval value higher than or equal to the time interval threshold defined by object 338Ah, in order to get the position capture validated as described below.



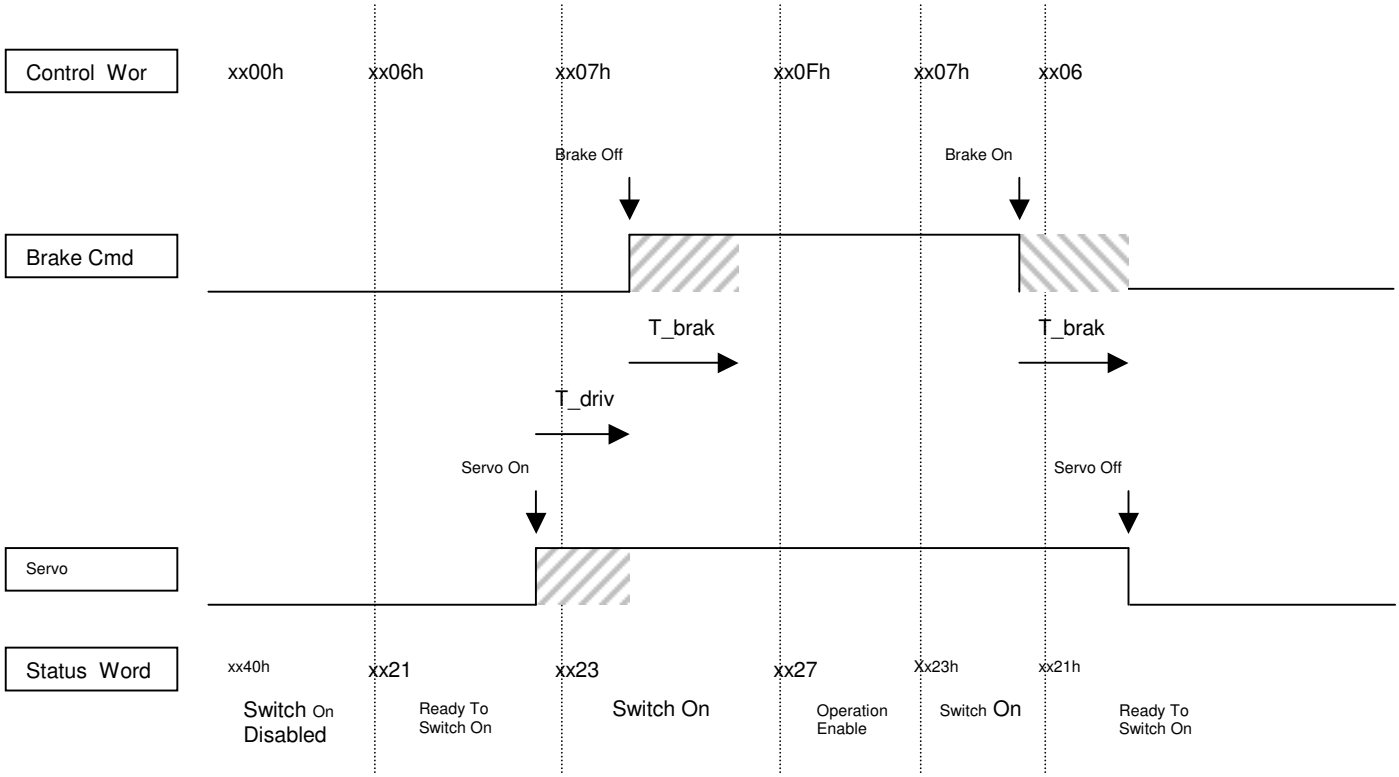
- **POSITION CAPTURE FILTER**

This parameter defines the "distance threshold" of the position capture filter. If the position gap between rising and falling edges is less than the threshold, then the signal is the following**:**



4.5. MOTOR BRAKE

Servo On/Off Timing Diagram



Index	Object	Name	Type	Attr.
0x3304	VAR	Amplifier Reaction Time (ms)	Unsigned16	rw
0x3305	VAR	Motor Brake Reaction Time (ms)	Unsigned16	rw

5. COMMISSIONING

5.1. MOTOR PARAMETERS

The motor parameters are stored in object 0x6410

These values are the parameters indicated in the manufacturer's catalogue.
The motor control parameters:
number of pole pairs (0x6410-13),
motor phase (0x6410-14),
motor offset (0x6410-15)
are respectively copied in objects 0x3410-1, 0x3410-2 and 0x3410-3.
Object 0x3410 can be possibly modified. It will be used for the motor control (e.g. if the resolver wiring or adjustment is not correct).
The autophasing procedure will calculate these parameters of object 0x3410.
The catalogue parameter of the motor inductance (0x6410-14) will be copied in object 0x340F-0 and will be used for the calculation of the current loop gains (0x60F6).

Object 0x340F-0 can be possibly modified before calculating the gains if inductances are serially mounted with the motor.
The catalogue parameter Maximum Motor Speed (0x6410-7) will clip the peak value of the motor speed in 0x6080.

5.2. CURRENT LIMITATIONS AND CURRENT LOOP ADJUSTMENT

The parameters defining the current limitation to be applied to the motor are the following:
- Motor Max. Current 0x6073
- Motor Rated Current 0x6075

The motor parameters Motor Max Current (0x6410-8) and (Motor stall Current) 0x6410-9 are used for the calculation of the internal drive limitations according to its maximum and rated currents (0x6510).
The values of the internal drive limitations can be displayed by means of object 0x30F4.
The current loop gains are accessible in object 0x60F6.
Object 0x3411 allows
- to calculate the current loop gains according to the motor parameters and drive specifications:
  Parameters:

  Inductance (0x340F)
  Drive Max. current (0x6510-1)
  Results:

  Current Loop Gains (0x60F6)

Object 0x3412 allows
- to calculate the drive current limitations according to the motor and drive currents (0x6510):
  Parameters:

  Motor Peak current (0x6410-9)
  Motor Stall current (0x6410-8)
  Drive Max. current (0x6510-1)
  Drive Rated current (0x6510-2)
  Results:

  Motor Max. current (0x6073-0)
  Motor Rated current (0x6075-0)

The input parameters must then be previously defined.

6. FILES

6.1. DEFINITIONS

6.1.1. OBJECT FILE FORMAT

The object file (i.e. CANopen object) is a plain text file format that allows the definition of an object list which values have to be set in the drive.
The syntax is: `index,sub=object_value`
All digital values can be hexadecimal (preceded by 0x) or decimal.
Just one single allocation per line is allowed.
A comment line starts with ";"
All lines that do not start with a figure will be ignored.

Example:
0x3549,10=0x12
means allocating value 0x12 to index object 0x3549, sub-index 10
13641,0xA=18
gives the same result..

6.1.2. CAM FILE

The object file can be used for defining the CAM table.
In this case, a CAM file can be the following:
; Auto-increment of the CAM table index
0x3549,1=0x8000
; First index for the CAM table
0x3549,2=0
; Last index for the CAM table (the table will have 1025 increments for 1024 segments)
`0x3549,3=1024`
; Define the current index in the table (in this example the first point of the table)
`0x3549,4=0`
; Enter the values in the table via object 0x3549,5
; The index will be automatically incremented
0x3549,5=0
0x3549,5=2
0x3549,5=7
0x3549,5=10
0x3549,5=12

Remarks
- The drive parameter file (DRIVEPAR.TXT) has the same format.
- The USER_PAR.TXT file is not mandatory. It allows, e.g., to define an initial configuration of the drive directly by the user.
- When loaded via an SD card, an object file can be directly executed in the memory without being saved in the GD1 drive flash memory.

6.1.3. USER PARAMETERS

6.2. FILES MANAGEMENT

6.2.1. FIRMWARE UPDATES

The various parts of the firmwares can be updated one by one.
There are two ways to update various firmwares in the drive:
-via a communication bus (CAN bus, RS-232 ...)
-via SD card
-via a communication bus (example: using the Gem Drive Studio file manager)

- **POWER CONTROLLER UPDATE**

Transfer the Power Controller firmware update file into the drive with the name PWUD0000.HEX
Start update procedure by writing 0x44555750 (PWUD) into object 0x5F50-1

- **CPLD UPDATE**

Transfer the CPLD update file into the drive with the name HWUD0000.BIN
Start update procedure by writing 0x44555748 (HWUD) into object 0x5F50-1

- **MAIN FIRMWARE UPDATE**

Transfer the firmware update file into the drive with the name FWUD0000.BIN
Start update procedure by writing 0x44555746 (FWUD) into object 0x5F50-1

6.3. SD Card

General
The GD1 allows file uploads from the SD flash memory card. This makes parameter file restoration or firmware updates possible.
Operation
The SD card must contain a text file named GD1LOAD.TXT and/or GD1RUN.TXT. These files contain a list of commands which will be interpreted **by the GD1.**

- **FILE LOADING SEQUENCE BY THE GD1:**

1. At power up:
   The DRIVEPAR.TXT file in the drive is loaded and executed (i.e. the saved parameters will be copied in the objects).

   If an SD card is already inserted, the GD1 will read the GD1LOAD.TXT and execute the commands in this file.

   The USER_PAR.TXT file in the drive is loaded and executed.
   The USERPROG.OUT file is loaded and executed.
   * The GD1 is already on:

   Each time an SD card is inserted, the GD1 reads the GD1RUN.TXT file and executes the commands in this file.

   After an operation with the SD card, check for no error (0x3022-2). If an error has occured, it must be analysed with the operation status.

Commands

The command shape in the GD1LOAD.TXT and GD1RUN.TXT is:
CMD PARAMETER
All lines beginning with ";" will not be executed (comment).

Supported commands:

Command	Parameters	Action
NODE	NodeID	If NodeID = 0 or NodeID is equal to the drive address, all following commands will be executed. If NodeID is different from the drive address, all following

		commands will be ignored.
LOAD	SD_FileName	Load the FileName file (contained in the SD card) into the GD1 buffer.
EXEC		Execute the formerly loaded file by LOAD. This is valid for object files only.
SAVE	GD1_FileName	Save the formerly loaded file by LOAD (i.e. in the GD1 buffer) in the GD1 flash memory with the name GD1_FileName.
DEL	GD1_FileName	Delete the GD1_FileName file from the GD1 flash memory.
FWUD		Update the firmware with the formerly loaded file by LOAD.
HWUD		Update the CPLD with the formerly loaded file by LOAD.
PWUD	 1	Update the PSoC with the formerly loaded file by LOAD. Update the PsoC in the drive with the PWUD0000.HEX file.
SPAR		Save the drive parameters in the DRIVEPAR.TXT file in the drive.
NOP	4 128	Do not load the USER_PAR.TXT file in the memory. Do not execute the user program tasks.

Remarks:

- The LOAD command only loads a file in the buffer and has therefore no effect on the drive operation. So, this command must be followed by at least on of EXEC / SAVE / FWUD / HWUD / PWUD commands. The buffer can contain only one single file.
- The NOP command avoids the loading of the USER_PAR.TXT file and/ or the user program tasks and has consequently effect only if used in the GD1LOAD.TXT file (see SD card operation). The user program tasks stopped can only be re-launched after switching on again the drive 24 V supply.
- The maximum number of commands per axis in a GD1RUN.TXT/GD1LOAD.TXT file is 32.

SD Card Status

If an error occurs while an SD card command is executed, the SD card process will be stopped at this line and object 0x5F20 allows to identify the error.

Index	Sub-index	Name	Type	Attr.	
0x5F20		SD Status	RECORD		
	0	Number of supported entries	Unsigned8	const	7
	1	Control	Unsigned16	rw	
	2	Error status	Unsigned16	ro	0 no error
	3	File Code	Unsigned16	ro	
	4	File Line	Unsigned16	ro	
	5	Command code	Unsigned16	ro	
	6	Error code	Unsigned32	ro	

Error Status:

Error	Description
0x0100	Card not supported (FAT32 format, type…)
0x0200	File not found
0x0300	File loading has failed
0x0400	File CRC error
0x0800	SAVE command error
0x0900	EXEC command error
0x0B00	LOAD command necessary
0x0C00	SD card reading error
0x0D00	File operation error (see File Error)
0x0E00	File type not correct
0x0A00	Update File identification error
0x0A10	Update File code error (FWUD, PWUD, HWPD…)

File Code: running file (or the latest executed file).

2 GD1LOAD.TXT
3 GD1RUN.TXT

File Line: number of the currently executed line (or of the latest executed line) in the GD1LOAD.TXT/GD1RUN.TXT file.
Command Code: code of the currently executed command (of of the latest executed command).

Code	SD Command
1	LOAD
2	SAVE
3	EXEC
4	NODE
7	FWUD
8	HWUD
9	PWUD
10	HDFD
11	SPAR
12	BIOS

Error Code: When executing an object file, this file may contain an error. In this case, the Error Code will contain the error code of the object file.

EXAMPLES:

Example 1: Restoration and update of the parameters for a 3-axis machine.
The SD card contains the parameter files PAR1.TXT, PAR2.TXT and PAR3.TXT for axes 1, 2 and 3.
The SD card must be successively inserted in all three drives for restoring the parameters of these axes.

- Content of the SD card:

GD1RUN.TXT
PAR1.TXT
PAR2.TXT
PAR3.TXT

* Content of the GD1RUN.TXT file:

```
; for Axis 1
NODE 1
LOAD PAR1.TXT
SAVE DRIVEPAR.TXT
EXEC
; for Axis 2
NODE 2
LOAD PAR2.TXT
```

```
SAVE DRIVEPAR.TXT
EXEC
; for Axis 3
NODE 3
LOAD PAR3.TXT
SAVE DRIVEPAR.TXT
EXEC
```

When the SD card is inserted in axis 1:
Axis 1 copies the PAR1.TXT file and saves it in the GD1 with the name DRIVEPAR.TXT, i.e. it replaces its parameter file.
Axis 1 then executes the PAR1.TXT file.
When the SD card is inserted in axis 2:
Axe 2 copies the PAR2.TXT file and saves it in the GD1 with the name DRIVEPAR.TXT, i.e. it replaces its parameter file.
Axe 2 then executes the PAR2.TXT file.
When the SD card is inserted in axis 3:
Axis 1 copies the PAR3.TXT file and saves it in the GD1 with the name DRIVEPAR.TXT, i.e. it replaces its parameter file.
Axis 1 then executes the PAR3.TXT file.

Notes:
PAR1.TXT, PAR2.TXT, PAR3.TXT are all object files.

Example 2: Restoration of the parameter files and loading different parameters in the axes.

- Load the parameter files (PAR1.TXT, PAR2.TXT, PAR3.TXT) for the 3 axes and save them in the drives.
- Load the specific parameters (CMD1.TXT, CMD2.TXT, CMD3.TXT) for each axis and the common parameters (CMD.TXT) to all 3 axes without saving.

Content of the SD card:

GD1RUN.TXT
PAR1.TXT
PAR2.TXT
PAR3.TXT
CMD1.TXT
CMD2.TXT
CMD3.TXT
CMD.TXT

Content of the GD1RUN.TXT file:

```
; for Axis 1
NODE 1
LOAD PAR1.TXT
SAVE DRIVEPAR.TXT
LOAD CMD1.TXT
EXEC
; for Axis 2
NODE 2
LOAD PAR2.TXT
SAVE DRIVEPAR.TXT
LOAD CMD2.TXT
EXEC
; for Axis 3
NODE 3
LOAD PAR3.TXT
SAVE DRIVEPAR.TXT
LOAD CMD3.TXT
EXEC
; for all axes
NODE 0
LOAD CMD.TXT
EXEC
```

When the SD card is inserted in axis 1:
Axis 1 copies the PAR1.TXT file and saves it in the GD1 with the name DRIVEPAR.TXT, i.e. it replaces its parameter file.
Axis 1 then executes the CMD1.TXT file (without saving) then CMD.TXT (without saving).
When the SD card is inserted in axis 2:
Axis 2 copies the PAR2.TXT file and saves it in the GD1 with the name DRIVEPAR.TXT, i.e. it replaces its parameter file.
Axis 2 then executes the CMD2.TXT file (without saving) then CMD.TXT (without saving).
When the SD card is inserted in axis 3:
Axis 3 copies the PAR3.TXT file and saves it in the GD1 with the name DRIVEPAR.TXT, i.e. it replaces its parameter file.
Axis 3 then executes the CMD3.TXT file (without saving) then CMD.TXT (without saving).

Notes:
PAR1.TXT, PAR2.TXT, PAR3.TXT, CMD1.TXT, CMD2.TXT, CMD3.TXT and CMD.TXT are all object files.
In the example above, the PAR1.TXT, PAR2.TXT and PAR3.TXT are not executed. The objects in these files will only be loaded at the next power up of the drives, with loading and execution of the DRIVE_PAR.TXT files.

Example 3: Update of the drive firmware.
Content of the SD card:

GD1RUN.TXT
FIRM0701.BIN

Content of the GD1RUN.TXT file:

```
; Firmware update
LOAD FIRM0701.BIN
FWUD
```

In order to avoid several updates if the SD card has been forgotten in the drive, the update must be made with GD1RUN.TXT only and never with GD1LOLAD.TXT.

REMARKS:
- The files must be located at the root of the SD card.
- The format of the file names must be 8.3.
- The Gem Drive can only read the SD card but cannot write in it. Preparation and copy of the files in the SD card must be made by means of a PC.
- The inserted card can only be taken into account at power off.

6.3.1. OBJECT FILE ERROR CODE

When executing an object file, an error may occur which error codes are the following:

File Error Code:

Error Code	Function
0xFFFFFFFF	A file with same name is already existing
0xFFFFFFFE	Access conflict: A file is already open
0xFFFFFFFD	Disk is full for writing
0xFFFFFFFC	File not found
0xFFFFFFFB	End of file
0xFFFFFFFA	File is not open
0xFFFFFFF9	Read error
0xFFFFFFF8	Write Error
0xFFFFFFF7	File is open
0xFFFFFFF6	Invalid file name
0xFFFFFFF5	File too large
0xFFFFFFF4	CRC32 control error

Object Error Code:

If the Error_Code value is higher than 16 bits, Error_Code is containing the index and sub-index of the error generating object.
bit 16-31: index
bit 8-15: sub-index

Compliances

- The SD cards used must be formatted in FAT12 or FAT16 (all standard SD cards with size up to 2 GB). FAT32 is not supported. When formatting under Windows, select the FAT format.
- The GD1 drives supplies the SD card with 3.5V.
- Micro SD and mini SD have not been tested.
- MMC cards are not supported (no detection signal when this card type is inserted).