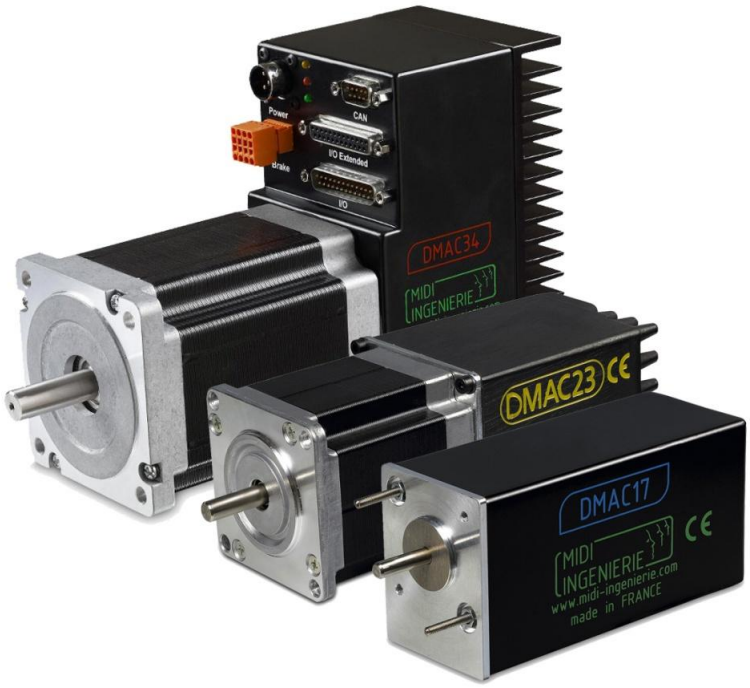
midi ingénierie
NEXEYA Products Division

3509, route de Baziège
31670 LABÈGE
France
Tél. : 33 (0)5 61 39 96 18
Fax : 33 (0)5 61 39 17 58
www.midi-ingenierie.com

midi ingenierie

DMAC17, DMAC23, DMAC34
User manual



Date : 23.11.12

Reference : dmac_v1_um_en.pdf
Ref. MI : CMN1480889_en.docx

Release : 1

Author : C.MARTY
http://www.midi-ingenierie.com

afaq ISO 9001 Qualité AFNOR CERTIFICATION

S.A.S. au capital social de 713 046€
333 373 108 RCS Toulouse
Siret 333 373 108 00013
APE 7112B
TVA FR60 333 373 108

GROUPE NEXEYA

Index

midi ingénierie
NEXEYA Products Division

1. Presentation

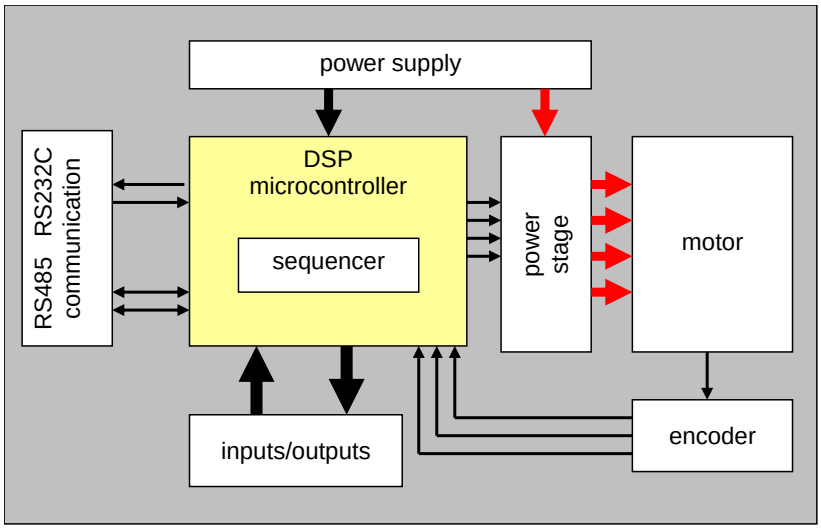
1.1. Introduction

DMAC module is a brushless motor integrated with its self-switched driver. Self-switched motor control guarantees the motor position at any time, avoiding motor stall. DMAC cumulates the advantages of stepper and brushless motor, making it suitable for both positioning, velocity or torque control.

DMAC is a compact module with simplified connection. It can be controlled using standard RS232 protocol (single-axis applications) or RS485 (multi-axis applications and higher noise immunity). CANopen DS402 (motion control) is available as an option.

1.2. Features

1.2.1. Architecture



1.2.2. Moves

Three kinds of movement can be performed:

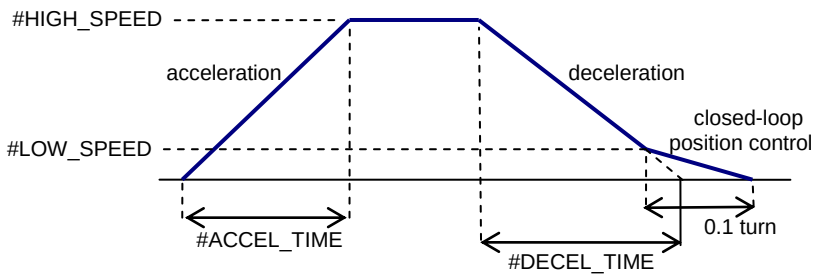
- Velocity mode (`MOVE_SPEED`) in which the motor turns at a constant speed.
- Position mode (`MOVE_TO` and `MOVE_ON`) in which the motor moves to a target position defined as absolute or relative.
- Interpolation mode (`MOVE_INTERPOL`) for synchronized multi-axis movement.

In Velocity and Position mode, trapezoidal or sinusoidal (S) velocity profile can be applied to optimize load acceleration and to fluidify movement. Acceleration and deceleration time can be adjusted by the user to suit every application.

In Position mode, the module performs closed-loop position control to reach target position with a precision of 1/10000th of a rev. This position control is performed at variable speed, proportionally to `#LOW_SPEED` and to position error.

A complete movement is thus decomposed into four distinct phases:

- Acceleration from current velocity (can be zero) to velocity setpoint.
- Constant speed movement.
- Deceleration from velocity setpoint to `#LOW_SPEED`.
- Position control from `#LOW_SPEED` to target position.



1.2.3. End-stops

DMAC has four end-stops:

- Two hardware end-stops using digital inputs IN1 and IN2 on DMAC17 and DMAC23 or IN9 and IN10 on DMAC34. User can connect sensors, dry contacts, etc…

- Two "virtual" software end-stops handled by the firmware, preventing the motor to go beyond `#POSITIVE_END` and `#NEGATIVE_END` positions. When enabled, if the motor tries to go beyond one of these end-points, movement is stopped and only reverse motion can be executed.

End-stops status is notified in variable `#STATUS`.

1.2.4. Sequencer

DMAC can record and execute up to 500 commands, allowing user to develop automation scripts that can be executed in standalone (without any computer or PLC).

Writing sequences is easy and intuitive. Memorized commands are executed at the rate of one per millisecond. Specific commands (`IF`, `JUMP`, `CALL`, `WAIT`, etc.) allow to control the sequence flow according to external events (digital and analog inputs, position or any internal variable).

1.2.1. Inputs / Outputs

DMAC17 and DMAC23 have 6 isolated digital inputs and 4 isolated digital outputs. DMAC34 has 8 isolated digital inputs and 8 isolated digital outputs. They can be used with the sequencer, for instance to launch predefined moves.

Additionally, DMAC34 has 2 non-isolated inputs dedicated to end-stops. Those two extra inputs can be used as regular inputs when end-stop feature is disabled.

One (DMAC17, DMAC23) or two (DMAC34) differential analog inputs can be used to connect sensors or potentiometers. Just like digital inputs, analog inputs can be used with the sequencer, for instance to control axis velocity or position with a potentiometer.

Additionally, DMAC34 offers two analog outputs to control external proportional actuators or give any information about the module.

1.2.2. Torque and Power

Maximum available torque can be adjusted using variable #TORQUE_RATIO up to:

- 0.5Nm for DMAC17
- 1.2Nm for DMAC23-1
- 2.2Nm for DMAC23-2
- 7Nm for DMAC34-1
- 10Nm for DMAC34-2

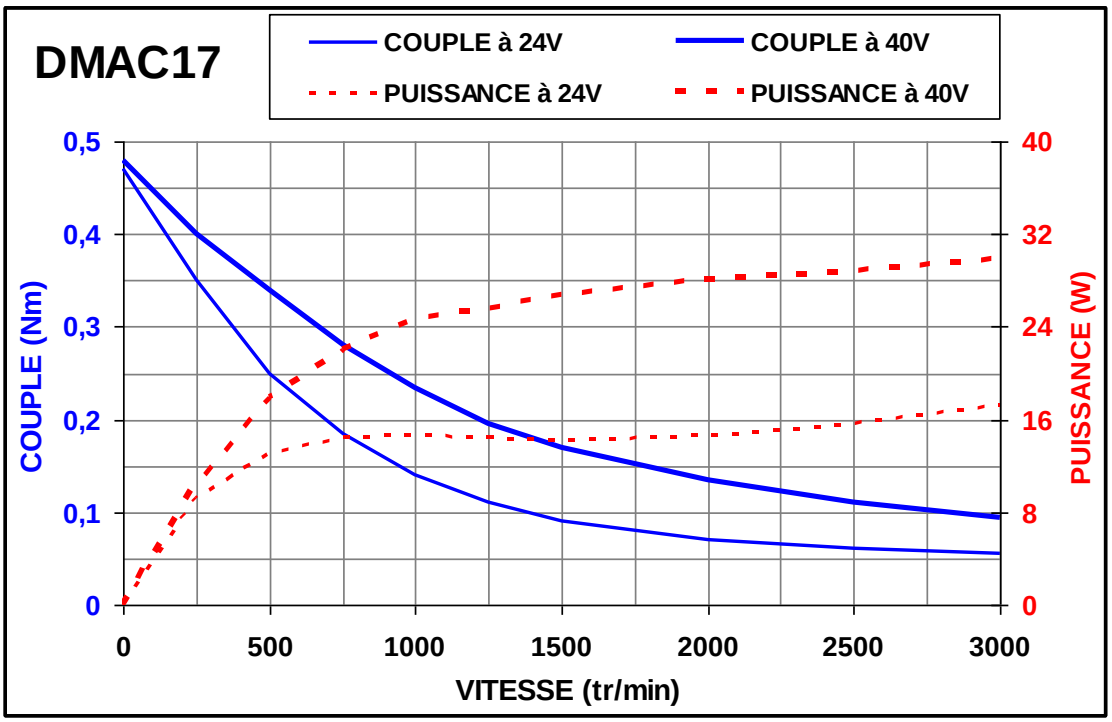
Special mode (`OPTIMIZED_CURRENT`) adjusts motor current to provide the necessary torque while minimizing heat losses.

Power can be enabled or disabled by software (`POWER ON/OFF`). It is automatically switched on at the beginning of a movement.

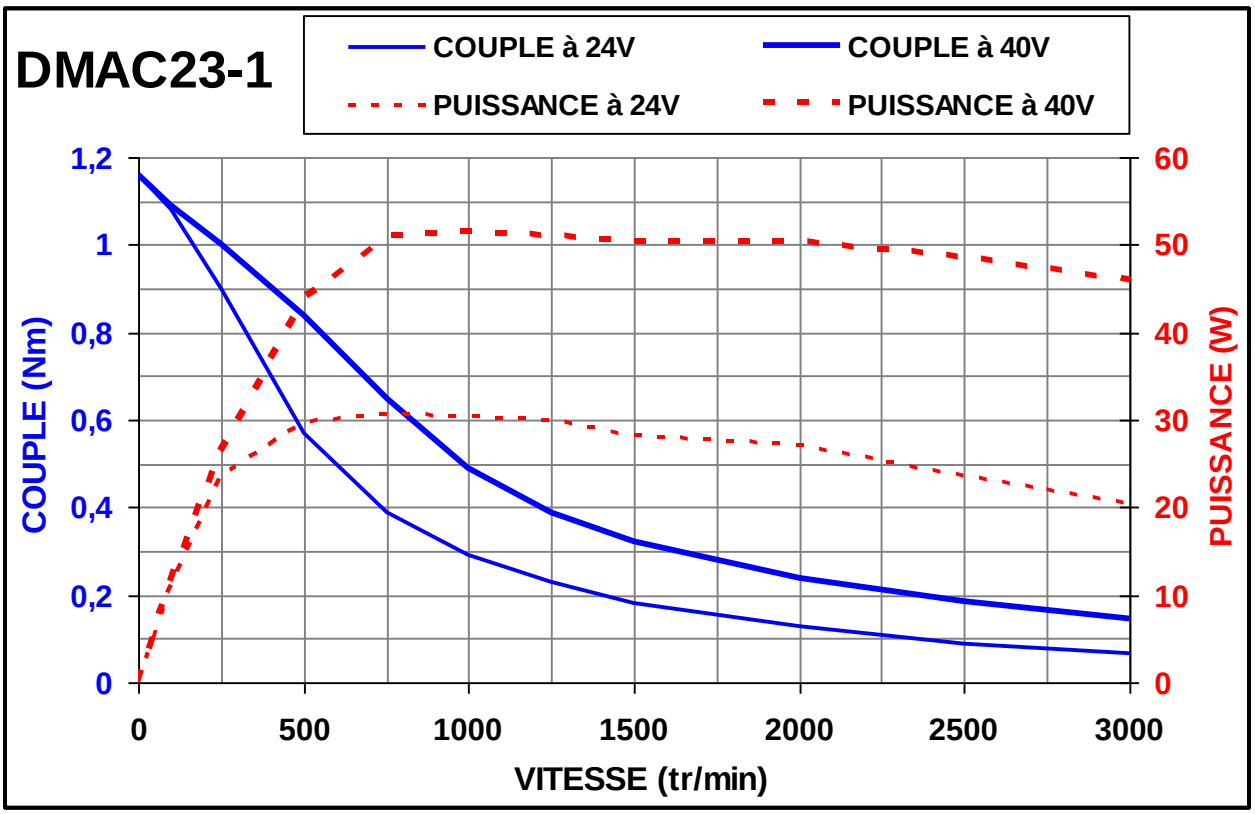
Motor torque is maximal at low speed and torque can even be applied at standstill for positioning applications.

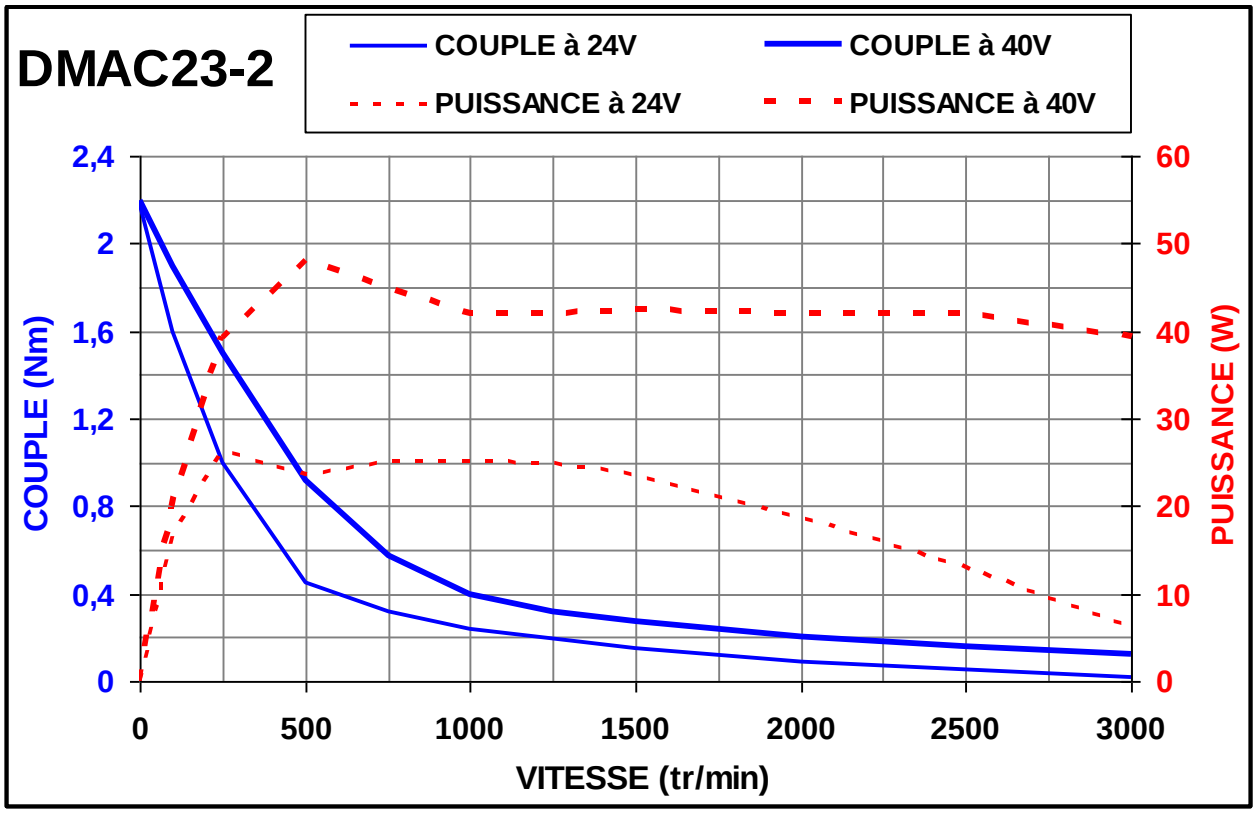
In the following figures:
COUPLE = TORQUE
VITESSE = SPEED
PUISSANCE = POWER
tr/min = RPM

DMAC17

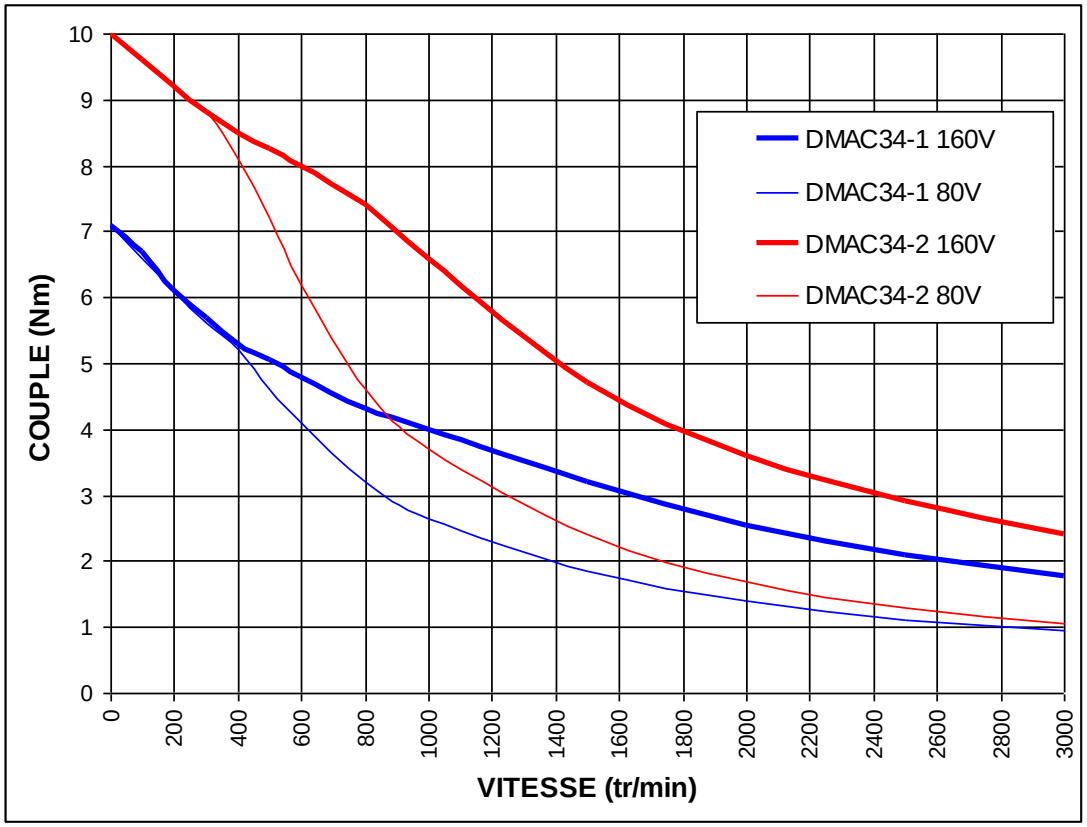


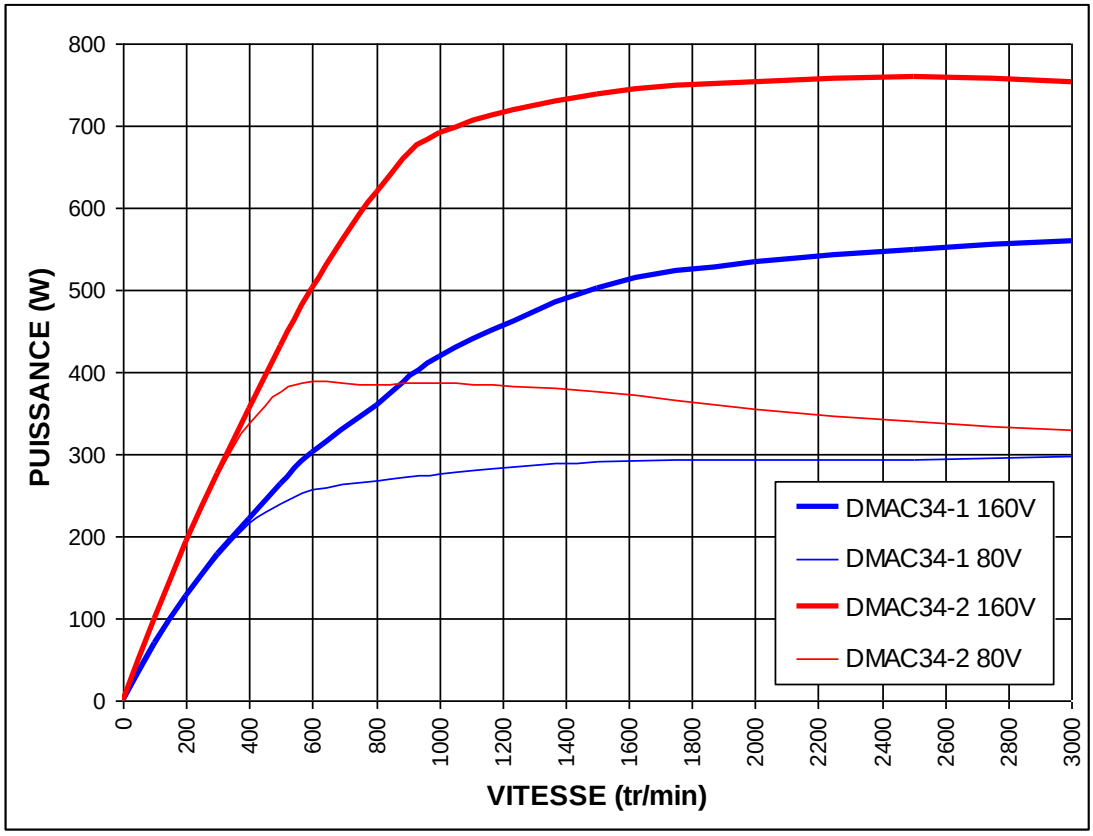
DMAC23


DMAC23-1
COUPLE à 24V
COUPLE à 40V
PUISSANCE à 24V
PUISSANCE à 40V
COUPLE (Nm)
PUISSANCE (W)
VITESSE (tr/min)
0
0,2
0,4
0,6
0,8
1
1,2
0
10
20
30
40
50
60
0
500
1000
1500
2000
2500
3000


DMAC23-2
COUPLE à 24V
COUPLE à 40V
PUISSANCE à 24V
PUISSANCE à 40V
COUPLE (Nm)
PUISSANCE (W)
VITESSE (tr/min)
0
0,4
0,8
1,2
1,6
2
2,4
0
10
20
30
40
50
60
0
500
1000
1500
2000
2500
3000

DMAC34


COUPLE (Nm)
VITESSE (tr/min)
DMAC34-1 160V
DMAC34-1 80V
DMAC34-2 160V
DMAC34-2 80V


PUISSANCE (W)
VITESSE (tr/min)
DMAC34-1 160V
DMAC34-1 80V
DMAC34-2 160V
DMAC34-2 80V

1.3. Safety guidelines

1.3.1. General rules

- Motors are IP30. Protect the motor from oil-mist, cutting oil, metal chips and paint fume, etc. Otherwise it may result in failure of electric circuits of the Driver Unit.
- Avoid projection of solvent, acids, bases.
- Avoid exposure to radiations.
- Do not remove the cover from a module. Internal voltage can be dangerous.
- Do not touch a powered module: risks of getting burnt or electric shock.
- Do not touch the motor shaft : risk of harm.
- Do not apply on the shaft of a DMAC34 an axial force ≥ 50N or a radial force ≥ 250N at 2 mm from the flange.
- Do not apply on the shaft of a DMAC17 or a DMAC23 an axial force ≥ 10N or a radial force ≥ 20N at 5 mm from the flange.

1.3.2. Storing

- Modules must be stored or moved in their original package or a suitable conditioning.
- Protect the modules from direct sunlight and humidity.
- Keep ambient temperature within –20°C to +40°C.

1.3.3. Proper use

- **Warning ! Motor temperature can reach 85°C. Do not touch the module or the motor.**
- **Always power off and wait at least 20s before servicing a module or its connectors.**
- Damages may occur if pinning specification is not strictly observed.
- The power supply shall be current-limited or a 5A (DMAC17 / DMAC23) or 10A (DMAC34) time-lag-fuse should be inserted in series between the supply and the module.
- The module shall not be installed in a confined enclosure and ambient temperature shall be kept between –10°C et +40°C.
- The cable shall not be submitted to repeated bending.
- The module shall be installed on a fixed, stable chassis, otherwise it may fall and be damaged or harm someone.
- Mechanical ground of the module should be connected to the mechanical ground of the chassis.
- Do not insert anything in the modules holes.

2. Technical specifications

2.1. Power supply

	Supply voltage	Max consumed current
DMAC17	12 to 45Vdc	2A
DMAC23	12 to 45Vdc	3A
DMAC34	20 to 160Vdc	9A

Warning! Consumed current increases as the supply voltage decreases.

Needed current depends on the total mechanical power needed: $P = T * \omega$ (P is the power in Watts, T is the torque in Nm, ω is the motor rotation speed in rad/s).

The higher the voltage, the higher is the high-speed torque.

Standstill or low-speed torque does not depend on the supply voltage.

If the voltage drops below 12V (DMAC17, DMAC23) or 20V (DMAC34), motion is stopped and the modules enter undervoltage lockout mode.

Note DMAC17 and DMAC23: when motor decelerates, kinetic energy is sent back to the power supply. The power supply must then be able to handle reverse current. Voltage can increase (output capacitor charging). DMAC17/23 have overvoltage detection that suspends deceleration/braking when supply voltage goes above 46V. Additional protection disables motor power if voltage goes above 49V. A fault is then notified and the module must be reset (power off or MODULE_RESET command) or the fault must be acknowledged (#ERROR:=0).

Whatever supply power is used, voltage can reach 49V with a DMAC17 or a DMAC23.

Optional ballast can dissipate extra energy in a resistor, so as not to reach the threshold.
If the supply voltage cannot handle nominal recovery voltage, one must insert a diode (75V/5A) in series between the module and the supply voltage. Such diode is integrated in DMAC terminal block.

Note DMAC34: when motor decelerates, kinetic energy is sent back to the power supply. The power supply must then be able to handle reverse current. Voltage can increase (output capacitor charging). DMAC34 have overvoltage detection that suspends deceleration/braking when supply voltage goes above 165V. Additional protection disables motor power if voltage goes above 170V. A fault is then notified (red led) and the module must be reset (power off or MODULE_RESET command) or the fault must be acknowledged (#ERROR:=0).

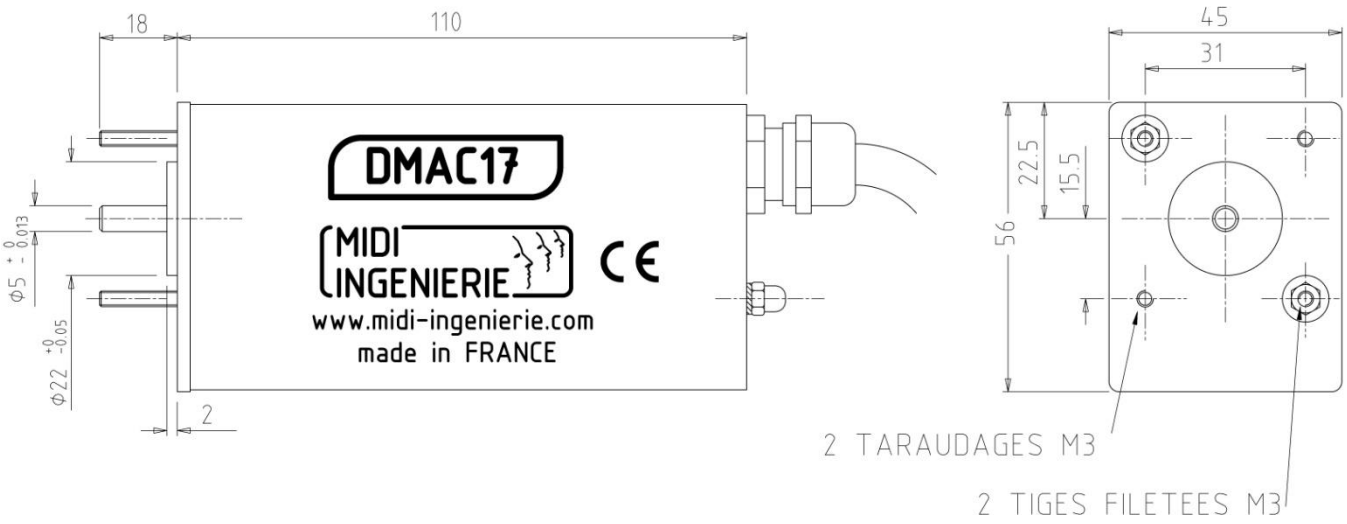
Whatever supply power is used, voltage can reach 170V with a DMAC34.

Optional ballast can dissipate extra energy in a resistor, so as not to reach the threshold.
If the supply voltage cannot handle nominal recovery voltage, one must insert a diode (200V/15A) in series between the module and the supply voltage.

2.2. Mechanical specification

2.2.1. DMAC17

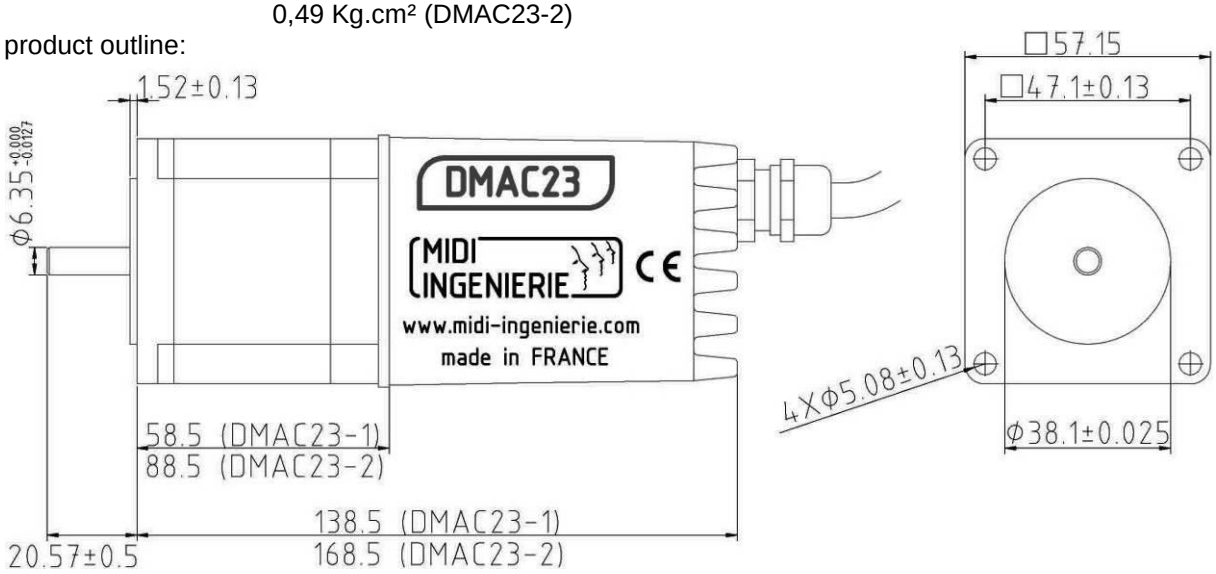
Size: 45mm x 56mm x 110mm (without shaft and cable)
Weight: 800g
Rotor inertia : 0,08 Kg.cm²
product outline:



2.2.2. DMAC23

Size:	57.15 x 57.15 x 138.5mm (DMAC23-1 without shaft and cable)
	57.15 x 57.15 x 168.5mm (DMAC23-2 without shaft and cable)
Weight:	1.5kg
Rotor inertia :	0,25 Kg.cm² (DMAC23-1)
	0,49 Kg.cm² (DMAC23-2)

product outline:



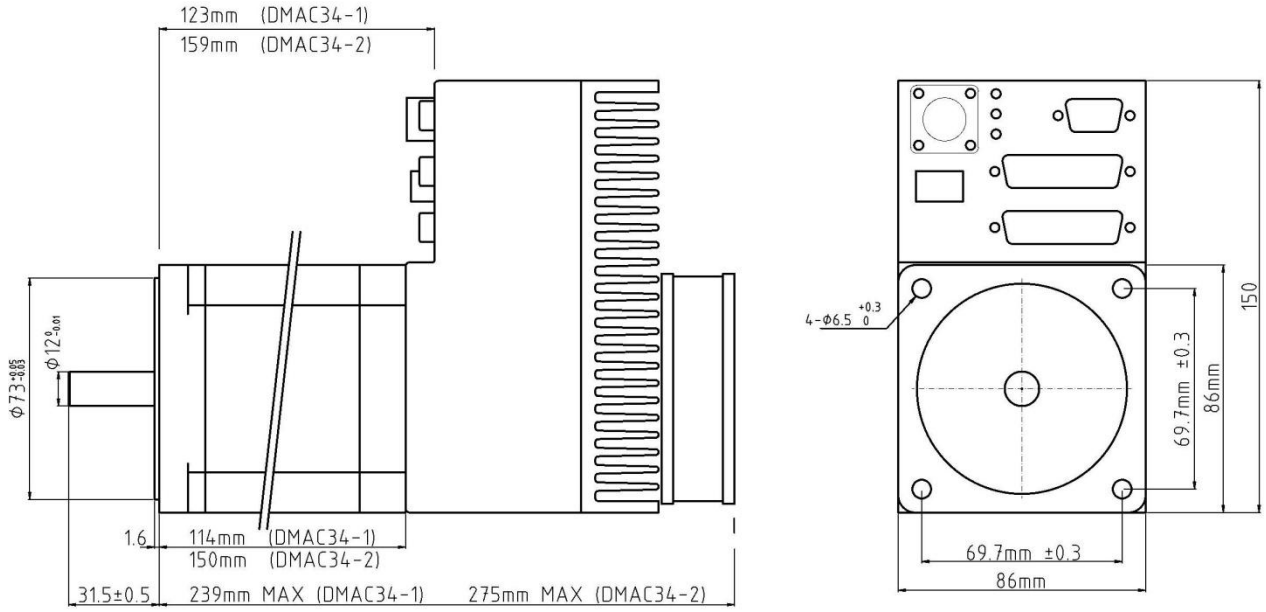
2.2.3. DMAC34

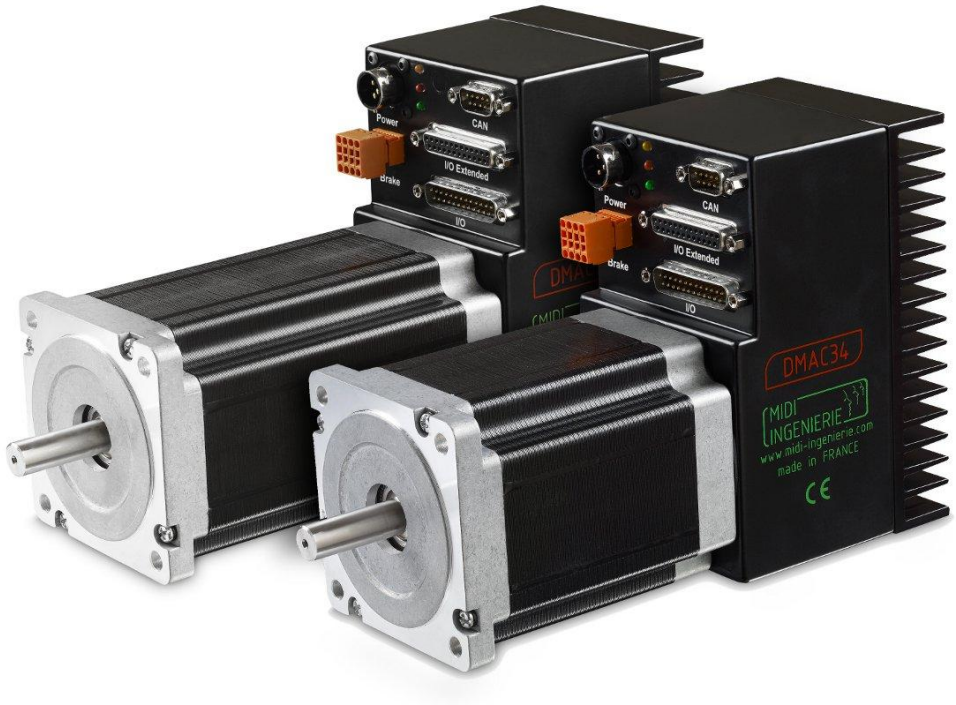
Size: 239 x 86 x 150mm (DMAC34-1 without shaft and cable)
275 x 86 x 150mm (DMAC34-2 without shaft and cable)

Weight: 4,8 Kg (DMAC34-1)
6,1 Kg (DMAC34-2)

Rotor inertia : 2,7 Kg.cm² (DMAC34-1)
4,05 Kg.cm² (DMAC34-2)

Product outline:

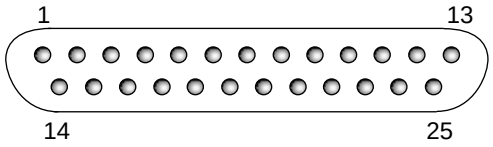




3. Pinning and configuration

3.1. DMAC17 connector description

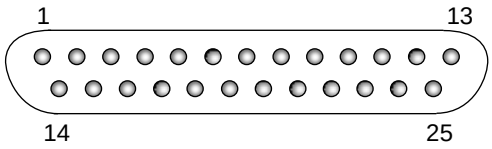
Pin	Description
1	Reserved
2	Communication RS485. /Z
3	Communication RS485. Z
4	Reserved
5	Communication RS485. 0V
6	Digital input 1 (IN1)
7	Digital input 2 (IN2)
8	Digital input 3 (IN3)
9	Digital input 4 (IN4)
10	Digital input 5 (IN5)
11	Digital input 6 (IN6)
12	Reserved
13	Reserved
14	Analog input (-)
15	Analog input (+)
16	Digital input ground (0V IO)
17	Outputs common + (+V IO)
18	Digital output 1 (OUT1)
19	Digital output 2 (OUT2)
20	Digital output 3 (OUT3)
21	Digital output 4 (OUT4)
22	Reserved
23	Reserved
24	0V supply
25	+V supply



DMAC17 Connector D-SUB male
(front view)

3.2. DMAC23 connector description

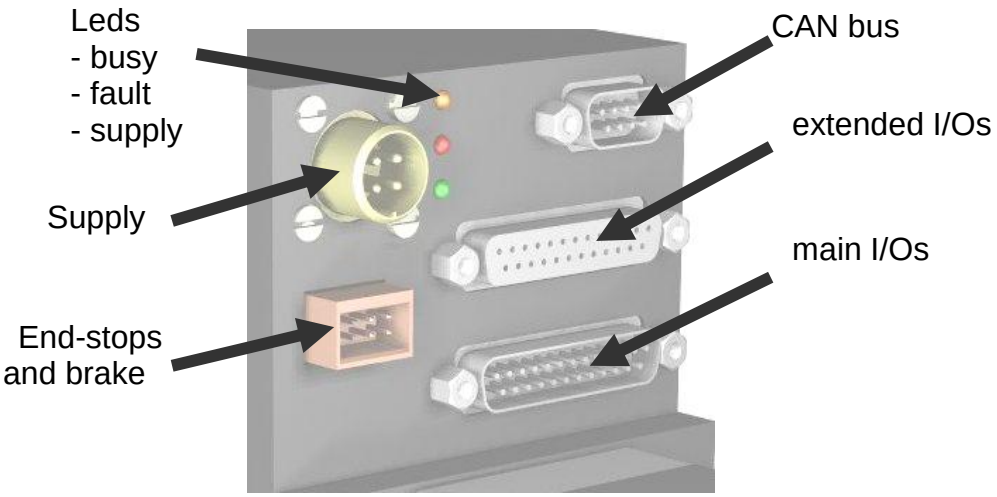
Pin	Description
1	Communication RS232. TXD
2	Communication RS485. /Z
3	Communication RS485. Z
4	Communication RS232. RXD
5	Communication 0V
6	Digital input 1 (IN1)
7	Digital input 2 (IN2)
8	Digital input 3 (IN3)
9	Digital input 4 (IN4)
10	Digital input 5 (IN5)
11	Digital input 6 (IN6)
12	Reserved
13	Reserved
14	Analog input (-)
15	Analog input (+)
16	Digital inputs ground (0V IO)
17	Digital outputs Common +V (+V IO)
18	Digital output 1 (OUT1)
19	Digital output 2 (OUT2)
20	Digital output 3 (OUT3)
21	Digital output 4 (OUT4)
22	Reserved
23	Reserved
24	0V supply
25	+V supply



MAC23 Connector D-SUB male
(front view)

3.3. DMAC34 connectors description

3.3.1. Global description DMAC34



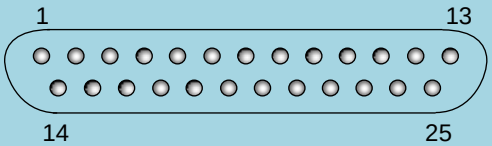
3.3.2. Supply connector

Circular plug 4 pts : power supply			
A	*+Vsupply*	**C**	*Mechanical gnd (shield)*
B	*0Vsupply*	**D**	*Reserved*

Compatible connectors (1 set included):
- Jack Free 4 ways (female)
  ref. TR1004PFS1NB (ITT CANNON)
  ref 1186713 (Farnell)
- Contact crimp female 16 gauge
  ref. 192991-0073 (ITT CANNON)
  ref 1186778 (Farnell)
- Backshell size 10
  ref. TR10ASR1N (ITT CANNON)
  ref 1188282 (Farnell)

3.3.3. Main I/Os connector

D-SUB 25 male : I/Os to DMAC terminal			
1	*TXD RS232*	***14***	*–IANA1*
2	*/Z RS485*	***15***	*+IANA1*
3	*Z RS485*	***16***	*0V IO*
4	*RXD RS232*	***17***	*+V IO*
5	*0V RS232_485*	***18***	*OUT1*
6	*IN1*	***19***	*OUT2*
7	*IN2*	***20***	*OUT3*
8	*IN3*	***21***	*OUT4*
9	*IN4*	***22***	*Reserved*
10	*IN5*	***23***	*Reserved*
11	*IN6*	***24***	*0Vaux*
12	*Reserved*	***25***	*+24Vaux*
13	*Reserved*		

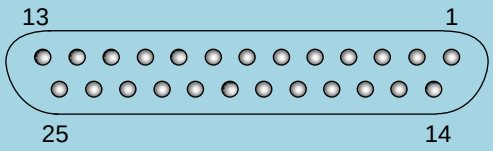


DMAC34 D-SUB 25 male (front view)

This connector is pin compatible with DMAC terminal. The terminal can be powered through +24V voltage (pin 24 an 25) if power consumption is below 2W. Otherwise, those two pins should not be connected and the terminal should be powered separately.

3.3.4. Extended I/Os connector

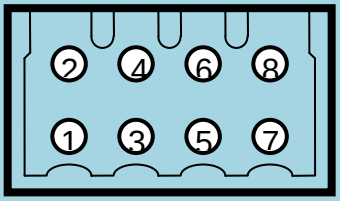
D-SUB 25 male : I/Os, extra functions			
1	*OUTA1*	**14**	*–IANA2*
2	*OUTA2*	**15**	*+IANA2*
3	*0Vana*	**16**	*0V IO*
4	*0Vana*	**17**	*+V IO*
5	*Reserved*	**18**	*OUT5*
6	*IN7*	**19**	*OUT6*
7	*IN8*	**20**	*OUT7*
8	*Reserved*	**21**	*OUT8*
9	*Reserved*	**22**	*COD A*
10	*COD /A*	**23**	*COD B*
11	*COD /B*	**24**	*0Vaux*
12	*0V_COD*	**25**	*+24Vaux*
13	*Reserved*		



D-SUB25 male (front view)

3.3.5. End-stops and brake connector

Tie-point block connector 8 pts : end-stops and brake			
1	*24Vaux*	**2**	*END+ (IN9)*
3	*0Vaux*	**4**	*24Vaux*
5	*END- (IN10)*	**6**	*0Vaux*
7	*+BRAKE*	**8**	*-BRAKE*

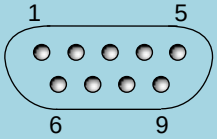


(front view)

Compatible connectors (1 set included):
Socket block 2 x 4 pts 3.5mm pitch female right
ref. B2L3.5/8 (Weidmuller)
ref 382-9602 (Radiospares)

3.3.6. CAN bus connector

D-SUB9 male : CAN bus					
1	*Reserved*	**4**	*Reserved*	**7**	*CANH*
2	*CANL*	**5**	*Shield*	**8**	*Reserved*
3	*0V CAN*	**6**	*Reserved*	**9**	*Reserved*



(front view)

3.4. Input/Output features

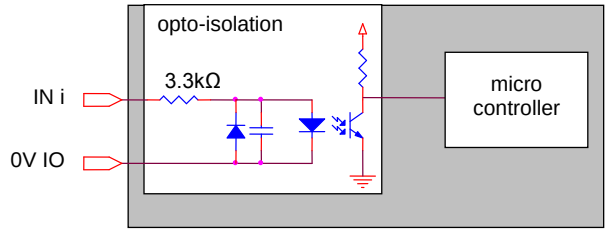
I/Os can be used as GPIOs (General Purpose Input or Output) or alternate functions.

	INPUT	FUNCTION
	IN1	GPIO or positive end-stop on DMAC17 and DMAC23
	IN2	GPIO or negative end-stop on DMAC17 and DMAC23
	IN3	GPIO
	IN4	GPIO
	IN5	GPIO and capture input or reference input
	IN6	GPIO ou Interpolation mode synchro
DMAC34 only	IN7	GPIO
DMAC34 only	IN8	GPIO
DMAC34 only	IN9	GPIO or positive end-stop input
DMAC34 only	IN10	GPIO or negative end-stop input

	OUTPUT	FUNCTION
	OUT1	BUSY or Interpolation sync or GPIO (according to `#OUTPUT_CONFIG.1`)
	OUT2	FAULT or GPIO (according to `#OUTPUT_CONFIG.2`)
	OUT3	GPIO
	OUT4	GPIO
DMAC34 only	OUT5	GPIO
DMAC34 only	OUT6	GPIO
DMAC34 only	OUT7	GPIO
DMAC34 only	OUT8	GPIO

3.5. I/Os specification

3.5.1. optically isolated digital inputs

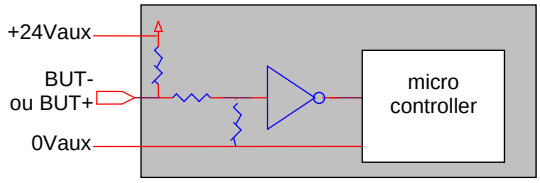


	min	max
VI_L (inactive) VI_H (active)	-30V 4V	1V +30V
II_L II_H	 1,1mA	25µA

Nominal voltage	5VDC to 24VDC
Signal voltage (inactive)	0VDC to 1VDC
Signal voltage (active)	4VDC to 30VDC
Admissible voltage	±30VDC
Galvanic insulation	50V

Reference voltage 0VIO is common for all inputs.
Digital inputs status can be read using `READ #INPUT`.

3.5.2. non-isolated digital inputs (DMAC34 end-stops)



	Min	max
VI_L (active) VI_H (inactive)	-0,3V +11V	+3V 30V

Nominal voltage 24VDC (internal pullup to 24V : current source 2mA)
Admissible voltage -0,3 to +30 VDC
Digital inputs status can be read using `READ #INPUT`.

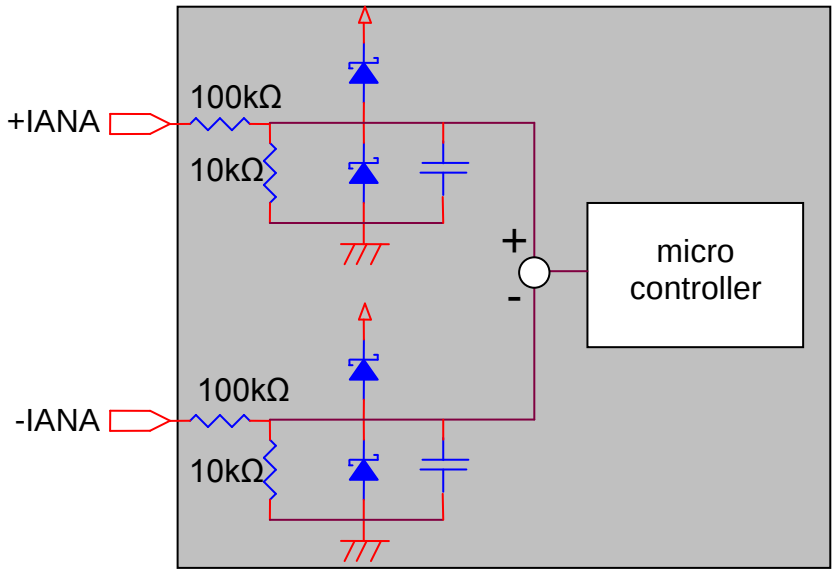
Non-isolated inputs are pulled-up to 24V and are inactive when left undriven. They are then active at electrical state 0V.

One can implement one of the following schemes:

- dry contact or optocoupler output directly connected between END signal and 0Vaux pin.
- 3 ways "NPN" sensor powered by pins +24Vaux/0Vaux and output connected to END
- 2-wire Proximity sensors powered by a current source. Sensors must match the following criteria: inactive current <0.5mA, active voltage < 3V

3.5.3. Analog inputs

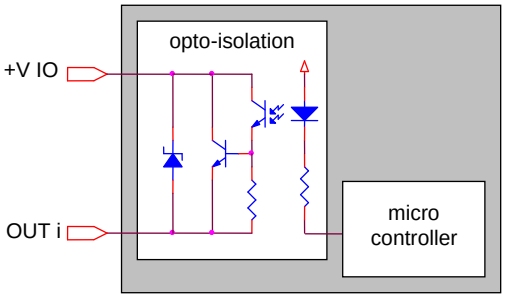
Functional diagram :



On DMAC17/23, variable #INPUT_ANALOG indicates differential voltage between pins +IANA and –IANA (expressed in mV)
On DMAC34, variable #INPUT_A1 indicates differential voltage betweens pins +IANA1 and –IANA1 (expressed in mV) and variable #INPUT_A2 indicates differential voltage between pins +IANA2 and –IANA2 (expressed in mV)

	DMAC17 /23	DMAC34
input voltage range / 0Vsupply	0 to 35V	±15V
differential input voltage range	±35V	±15V
Cutoff frequency	1KHz	1KHz
Differential input impedance	220KOhms	220KOhms
Precision	±3%	±3%
Resolution	8,6mV	7,3mV

3.5.4. Optically isolated outputs

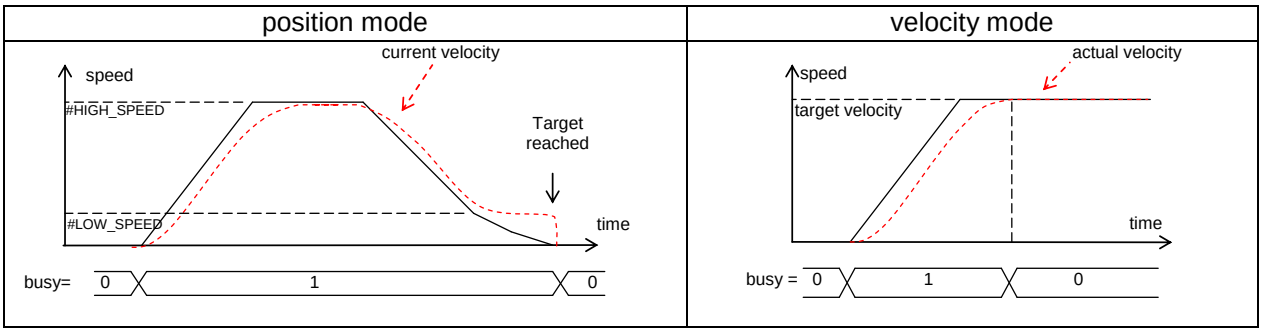


	min	max
Io_{off}		0,1mA
Vo_{on}		0,6V @1mA 1,1V @5mA 1,3V @50mA

Max output current 50mA
Max output voltage 30V
+VIO voltage is common to all digital outputs.
Logical 1 (`#OUTPUT:=1`) optocoupler is ON (driving).
Logical 0 (`#OUTPUT:=0`) optocoupler is OFF (released).

Logical output can be set using #OUTPUT and read using READ #OUTPUT

OUT1 can reflect BUSY signal if corresponding bit of #OUTPUT_CONFIG is set to 1. BUSY signal is active as long as target is not reached (target velocity in velocity mode or target position in position mode). See diagram:



OUT2 can reflect FAULT signal if corresponding bit of #OUTPUT_CONFIG is set to 1. FAULT signal is active when one of the following events occurs:
Motor overcurrent disjunction
Overvoltage disjunction
Undervoltage disjunction
Thermal disjunction (motor or electronics above 85°C or CPU above 120°C)

3.5.5. Analog outputs (DMAC34 only)

Analog outputs OUTA1 and OUTA2 are referred to 0Vana (non-isolated).
Desired output voltage is expressed in mV :
`00#OUTPUT_A1 := 4520` ; forces 4.52V on analog output 1
`00#OUTPUT_A2 := 2125` ; forces 2.125V on analog output 2

Output voltage range :	0 to +10V
Cutoff frequency:	1kHz
Resolution :	2,6mV
Precision :	2%
Max load :	0,1 mA
Max admissible current :	-1mA < I_0 < 1 mA

3.5.6. Brake output (DMAC34 only)

This non-isolated output (+BRAKE, -BRAKE) can drive a failsafe brake. Command BRAKE ON disables current supply thus enabling the brake. Command BRAKE OFF powers the brake under 24V, thus releasing motor axis.

Nominal brake voltage :	24VDC
Max current :	600mA

One must observe brake polarity and connect a diode in anti-parallel of the brake in order to avoid any problem should the brake be accidentally disconnected.
An optional failsafe brake can be adapted in front of the DMAC34.

3.5.7. Encoder output (DMAC34 only)

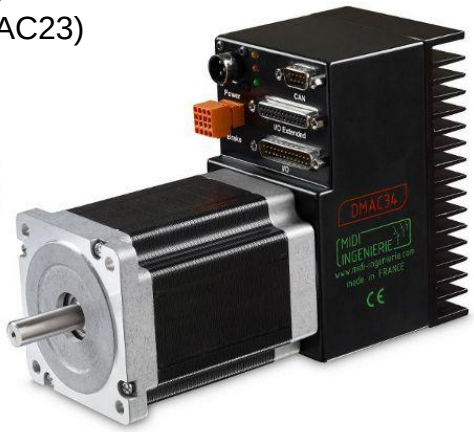
This output reproduces incremental 500pts/rev. encoder signals.
It is differential RS422 output with signals CODA, COD/A, CODB, COD/B referred to 0V_COD (non-isolated).

3.5.8. I/Os cabling sample

DMAC Terminal

25pts shielded cable
(integrated in DMAC17/DMAC23)

DMAC34



Note: DMAC Terminal is directly powered via DMAC34 +24Vaux voltage. One should not connect a power supply to the terminal.

3.6. Visualization (DMAC34 only)

DMAC34 status is summarized by 3 leds :

- Yellow "Busy" led shows motor activity: current motion or sequence.
- Red "Fault" led indicates fault state: overvoltage, undervoltage, thermal disjunction or motor disjunction. Moreover, this led flashes at module power-up or reset.
- Green "Power" led shows that DMAC34 is powered-up. Warning: the led does not indicates that supply voltage is inside the tolerance range (it could either be too low or too high).

3.7. Configuring COM port

DMAC modules implement both RS232 and RS485. RS232 protocol is to be used with single-axis applications in a low-noise environment. RS485 protocol is to be used with multi-axis applications and offers better noise immunity.

3.7.1. Setting-up the module

DMAC should be setup with the following parameters:
`SET_BAUDRATE` (transfer speed in bauds 9600, 19200, 38400 or 115200)
`SET_ADDRESS` (module address 0 to 63)
`#LINE_DELAY` (RS485 turn-around delay in microseconds, 100 to 3000)

Factory default values are:

```
SET_BAUDRATE 38400
SET_ADDRESS 0
```

```
#LINE_DELAY:=3000
```

3.7.2. RS-485 protocol

What is RS485?

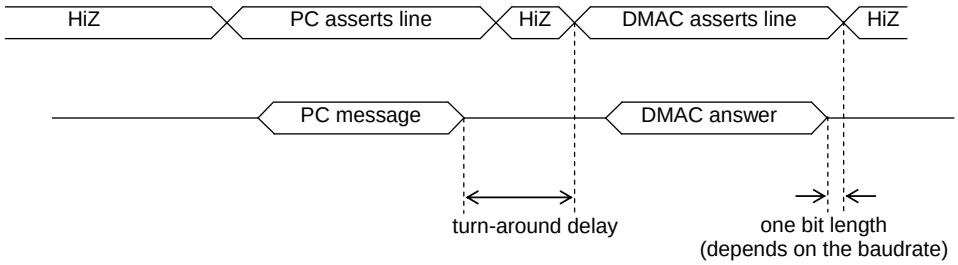
RS-485 allows multiple devices (up to 32) to communicate at half-duplex on a single pair of wires, plus a ground wire, at distances up to 1200 meters (Both the length of the network and the number of nodes can easily be extended using repeaters). Different modules types (DMAC 17/23 or 34) can be connected on the same network, providing they all use the same baudrate and distinct addresses.

How does the hardware work?

Data is transmitted differentially on two wires twisted together, referred to as a "twisted pair", which provides the modules with high noise immunity and long distance. Emitters and receivers are all connected on the same bus. Only one device can drive the line at a time, so drivers must be put into a high-impedance mode (tri-state) when they are not in use.

How does the software work?

The half-duplex tri-state mode necessitates insertion of a delay between asserting the line and transmitting the data on the line, as well as between end of transmission and line release. Those delays are handled by DMAC firmware and control software (PC or PLC). Line handling is directly controlled by software when using drvMI dll or Winsim2.



How to communicate with the module?

WinSim2 Windows-based GUI application for DMAC and Midi Ingenierie products. Offers various visualization and control tools.
DrvMi.dll Dynamic library. Can be used within user applications (C/C++, VisualBasic, Delphi, etc…)

4. Commands

Command frames are sent to the module via a character string composed of the address, the command and parameters.

Entire command name (e.g.: `MOVE_TO`) or its mnemonic (e.g.: `MTO`) may be used.

Module address is given by the first two characters of the frame.
If the address is omitted, the command is "global" and shall be executed by all the modules. It is acknowledged only by the module at address 0.

Several commands can be sent within a single frame, separated by comas
(e.g.: "`#HIGH_SPEED:=10000, MOVE_TO 1234`").
Total frame length should not exceed 256 characters.

A space character should be inserted between command and parameter.

Some commands can be used only within a sequence or only in "live" (not within a sequence). Some others can be used in both modes.

In this manual, the following symbols are used:
@ represents module address
[ ] represents optional parameter

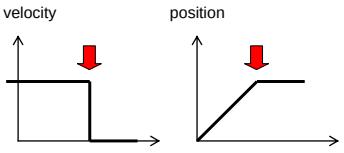
4.1. BRAKE (DMAC34 only)

Syntax:	[@]BRAKE parameter
Mnemonic:	BRA
Parameter:	ON, OFF
Description:	Drives the (optional) fail-safe brake. `BRAKE ON` activates the brake, preventing any movement. `BRAKE OFF` powers the brake, thus releasing the axis.
Example:	`08BRAKE ON` Activates braking on module 08.

4.2. CALL / RETURN

Syntax:	CALL parameter RETURN
Mnemonic:	CAL RET
Parameters:	number of the first line of the function 1 < parameter < 500
Description:	Call of a subroutine. A subroutine must end with the RETURN command. Sequence then jumps back to line following CALL command.
Note:	Up to 5 subroutines calls can be nested.
Example:	`:4 CALL 12` Call of the subroutine at line 12 `NOP` `NOP` `:12 NOP` Line 12: start of the subroutine `NOP` `RETURN` Returns to line 5 (following CALL command)

4.3. HALT

Syntax:	[@]HALT [parameter]
Mnemonic:	HAL
Parameter:	None, MOUV or SEQ
Description:	Module stops without deceleration ramp. Braking torque is maximum, deceleration is determined by holding torque and load inertia. Position tracking is still active.  HALT command without any parameter stops both movement and sequence. HALT MOUV stops only movement but keeps sequence running. HALT SEQ stops the sequencer but has no effect on the movement.
Example:	`05HALT` Axis 05 stops immediately. If a sequence is running, it is also stopped.

4.4. HARD_ENDS

Syntax:	[@]HARD_ENDS parameter
Mnemonic:	HEN
Parameter:	ALL (enables hardware end-stops) or OFF (disables hardware end-stops) POS (enables positive end-stop only) NEG (enables negative end-stop only)
Description:	Enables or disables hardware end-stops. Positive end-stop stops and prevents forward (CW) motion. Negative end-stop stops and prevents backward (CCW) motion. **This parameter is automatically stored at shut-down.**
Warning:	Polarity inversion (INVERSE_POLARITY command) applies to hard-ends input as well as regular inputs.
Example:	`03HARD_ENDS ALL` Enables hardware end-stops on module 03.

4.5. IF

Syntax:	IF test JUMP parameter IF test CALL parameter IF test JUMP_REL parameter
Mnemonic:	IF
Parameters:	"test" is a logical operation having TRUE or FALSE result. "parameter" is the line number to jump to if test result is TRUE (otherwise, sequencer goes on incrementing line number). [0 ; 500]
Description:	Conditional jump. Linear execution of the sequence is disrupted. Jump type can be one of `JUMP`, `CALL` or `JUMP_REL` (see commands details)
Example:	`:10 IF #POSITION > 1234 JUMP 56` If current position is greater than 1234, Sequencer will execute line 56. Otherwise, execution will continue at line 11.

4.6. INVERSE_POLARITY

Syntax:	[@]INVERSE_POLARITY parameter
Mnemonic:	IPO
Parameter:	OFF (inputs and outputs are not inverted) ALL (inputs and outputs are inverted) IN (inputs are inverted) OUT (outputs are inverted)
Description:	Specifies if physical state is inverted vs logical state of `#INPUT` et `#OUTPUT` variables. Inputs: Standard: active physical state ⇔ `#INPUT` is 1 Inverted: active physical state⇔ `#INPUT` is 0 Outputs: Standard: `#OUTPUT` is 1 ⇔ active physical state Inverted: `#OUTPUT` is 0 ⇔ active physical state
Example:	`03INVERSE_POLARITY OUT` From now on, physical ouputs will be inverted vs `#OUTPUT`.

4.7. JUMP / JUMP_REL

Syntax:	JUMP param1 JUMP_REL param2
Mnemonic:	JUM JRE
Parameter:	0 <= param1 < 500 line number to execute (JUMP) -500 < param2 < 500 line offset to jump to (JUMP_REL)
Description:	Sequence line jump. Linear execution is disrupted to jump to a given line. `JUMP` command jumps to a given line whatever the current line is, whereas `JUMP_REL` jumps to an offset from current line. `JUMP_REL` can be either positive or negative (backward jump).
Note:	`JUMP 0` blocks program execution.
Example:	`JUMP 10` Jump to line 10 `:15 JUMP_REL -1` Loop to previous line (line 14). `JUMP_REL +2` Jumps two lines forward

4.8. MODULE_RESET

Syntax:	[@]MODULE_RESET [parameter]
Mnemonic:	MRE
Parameter:	None or ALL (reset factory settings).

Description:	Motor is powered off and then stopped (short-circuited) until complete stop.
	MODULE_RESET command is then similar to power-off/ power-on.
	Outputs are forced inactive. Volatile variables are (#V1 to #V32) are set to 0. If a sequence was configured to be executed at reset, it is launched.
	
	Parameter ALL restores factory settings (except serial line configuration):
	`#HIGH_SPEED := 60000`
	`#LOW_SPEED := 6000`
	`#TORQUE_RATIO := 50`
	`#ACCEL_TIME := 1000`
	`#DECEL_TIME := 1000`
	`#POSITIVE_END := +10000`
	`#NEGATIVE_END := -10000`
	`#POSITION := 0`
	`#OUTPUT_CONFIG := 3`
	`#ON_RESET := 0`
	Memorized variables (#M1 to #M8) are set to 0.
	Standard I/Os polarity is restored (`INVERSE_POLARITY OFF`).
	End-stops are disabled (`HARD_ENDS OFF; SOFT_ENDS OFF`).
	Motion configuration is restored `(S_CURVE OFF,OPTIMIZED_CURRENT OFF).`
	
	Note : Serial line configuration (baudrate, address and `#LINE_DELAY`) are not affected by MODULE_RESET ALL command

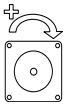
Example:	`02MODULE_RESET` Resets module 2.

4.9. MOVE_INTERPOL

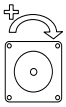
Syntaxe	[@]MOVE_INTERPOL parameter
Mnemonic:	MIN
Parameter:	Position offset (relative) (in motor increments: 10^{-4} rev.) to be executed during `#INTERPOL_TIME` . [-2 147 483 648; +2 147 483 647]
Description:	In interpolation mode, each axis describes a trajectory defined by a succession of "segments". A segment is defined by the position offset to be executed in a given time period (`#INTERPOL_TIME`). At each `MOVE_INTERPOL` command, one segment is stored in a FIFO buffer (which size is defined by `#INTERPOL_FIFOSIZE` variable) Movement is launched by global command "`SYNCHRO INTERPOL`". All axes execute previously stored segments in a synchronized way. (see `#INTERPOL_MODE` variable for details on multi-axis synchronization) When FIFO is full, `MOVE_INTERPOL` command is rejected by emission of XON_ERROR (17h) character. When FIFO is empty (the entire trajectory has been described) movement is stopped and "`SYNCHRO INTERPOL`" command is necessary to start another interpolated movement. During the movement, user is in charge of supplying the modules with segments at a higher rate than trajectory execution.
Note:	MIShell (PC application) can handle automatic command re-send in case of FIFO-full error. To enable this feature, one should add underscore character before the command: `_01MOVE_INTERPOL 200`
Example:	Sample commands for 3-axis interpolation:

```
#INTERPOL_FIFOSIZE:=64    ;FIFO contains 64 segments
#INTERPOL_TIME:=100       ;one segment per 100ms
#INTERPOL_MODE:=0         ;SimpleSYNC mode

00MOVE_INTERPOL 100      ;1st segment
01MOVE_INTERPOL 750      ;(stored in FIFO but not
02MOVE_INTERPOL 100      ;launched yet)

SYNCHRO INTERPOL         ;synchronized start

00MOVE_INTERPOL 100      ;2nd segment
01MOVE_INTERPOL -50
02MOVE_INTERPOL 100

…                         ;3rd segment…
```

4.10. MOVE_ON

Syntax:	[@]MOVE_ON parameter
Mnemonic:	MON
Parameter:	Position offset in motor increments (10^{-4}rev.) [-2 147 483 648 ; + 2 147 483 647]
Description:	Motor rotates by a given amount of increments (position offset). Motion follows pre-defined velocity profile (`#ACCEL_TIME`, `#DECEL_TIME` and `#HIGH_SPEED` variables). Movement direction is defined by parameter sign, positive parameters resulting in clockwise rotation (front view). 
Note:	`MOVE_ON` forces motor power ON.
Example:	`03MOVE_TO -5000` Module 03 moves by half a rev. counter-clockwise

4.11. MOVE_SPEED

Syntax:	[@]MOVE_SPEED parameter
Mnemonic:	MSP
Parameter:	Target velocity expressed in 0.01RPM. [-400000 ; 400000]
Description:	Motor rotates at a given speed. Movement direction is defines by parameter sign, positive parameters resulting in clockwise rotation (front view). Velocity profile goes from current velocity to target velocity in a time defined proportionally to `#ACCEL_TIME and #DECEL_TIME variables`.
Note:	 Target velocity is limited (in absolute value) to `#HIGH_SPEED variable`. Parameters superior to `#HIGH_SPEED` will be executed at `#HIGH_SPEED` velocity. `MOVE_SPEED` forces motor power ON.
Example:	`00MOVE_SPEED 50000` Module 00 rotates at 500RPM clockwise.

4.12. MOVE_TO

Syntax:	[@]MOVE_TO parameter
Mnemonic:	MTO
Parameter:	Target position (10^{-4}rev.) [-2 147 483 648 ; + 2 147 483 647]
Description:	Motor moves from current position to target position. Velocity profile is defined by `#ACCEL_TIME`, `#DECEL_TIME` and `#HIGH_SPEED variables`.
Note:	If motor is already at target position, no movement is performed. This command forces motor power-on.
Example:	`00MOVE_TO 123456` Motor 00 goes to position +123456

4.13. OPEN_SEQ / CLOSE_SEQ

Syntax:	[@]OPEN_SEQ [@]CLOSE_SEQ
Mnemonic:	OSE CSE
Description:	OPEN_SEQ enable sequence edition. CLOSE_SEQ disables sequence edition. Bit STATUS.16 is set to 1 to indicate that sequence edition is enabled. In sequence edition, commands are not executed but stored in the module for future execution. Sequence starts at line number 1 (default) but user can force edition of a given sequence line by preceding command by ":n" (n being the sequence line number) `OPEN_SEQ` command erases any previously memorized sequence. Sequence edition remains enabled until `CLOSE_SEQ` command is sent to the module.
Example:	`OPEN_SEQ` Enable Edit mode. `NOP` `NOP` Commands are memorized in the sequencer `NOP` `CLOSE_SEQ` Disable Edit mode

4.14. OPTIMIZED_CURRENT

Syntax:	[@]OPTIMIZED_CURRENT parameter
Mnemonic:	OCU
Parameter:	ON (optimized mode) or OFF (default mode)
Description:	Enables or disables "optimized current" feature. When enabled, motor current is automatically adjusted to provide the necessary torque. This feature lessens thermal losses when motor does not run continuously. **This parameter is automatically stored at shut-down.**
Example:	`06OPTIMIZED_CURRENT ON` Enables optimized current feature for module 06.

4.15. POWER

Syntax:	[@]POWER parameter
Mnemonic:	POW
Parameter:	ON,OFF,SC (short-circuiting motor coils)
Description:	POWER ON applies power to motor coils. POWER OFF implies no current and no torque. POWER SC generates low-speed resisting torque.
Note:	`POWER ON` is automatically performed before any MOVE command.
Example:	`02POWER ON` Enable power for module 2.

4.16. READ

Syntax:	@READ parameter
Mnemonic:	REA
Parameter:	Variable name (variables are described next)
Description:	Sends variable value over serial line toward PC (or PLC). Use 'H' prefix in front of variable name to read hexadecimal value. Use 'B' prefix in front of variable name to read binary value. H et B can either be capital letters or small characters.
Note:	Hexadecimal and binary values are 4 bytes signed. `READ` command cannot be used within a sequence. Character string sent back by the module is composed of the module address (2 chars), the mnemonic of the variable, the '=' character and the value of the variable.

Example:	`00READ #POSITION`
	reads module 00 current position:
	module sends: `00#POS=12345`
	
	`01READ h#OUTPUT`
	reads module 01 digital outputs (hexadecimal format):
	module sends: `01#OUT=h00000007`
	=> Outputs OUT1, OUT2 and OUT3 are active, OUT4 is inactive (provided `INVERSE_POLARITY` is OFF).
	
	`02READ b#INPUT`
	reads module 02 digital inputs (binary format):
	module sends: `02#INP=b00000000 00000000 00000000 00001010`
	=> Inputs IN2 and IN4 are active, inputs IN1, IN3,IN5 are IN6 inactive (depending on `INVERSE_POLARITY`).

4.17. READ_SEQ

Syntax:	[@]READ_SEQ parameter
Mnemonic:	RSE
Parameter:	address of sequence line to read [1 ; 500]
Description:	Reads one line of a sequence. Useful for sequence programming. Warning, syntax can be slightly different from the one used in sequence edition.
Example:	`00READ_SEQ 3` Reads line 3 Module sends: `00:003 MTO +2000` Line 3 contains command "`MOVE_TO 2000`"

4.18. REFERENCE

Syntax:	[@]REFERENCE parameter
Mnemonic:	REF
Parameter:	ON (enabled) or OFF (disabled)
Description:	Enable/disable "reference mode". Reset position: Position is set to zero if motor is running clockwise and a rising edge (0 to 1) is detected on IN5 input. Reference mode is automatically disabled once position has been initialized.
Example:	`06REFERENCE ON` Enable reference mode on module 6.

4.19. REQUEST_VERSION

Syntax: [@]REQUEST_VERSION

Mnemonic: RV or RVE

Description : Reads module identification string.
Module answers :
@EV vV.RR *CODE* ‘‘MIDI-INGENIERIE*_product_serial_manufacturing-date_revision-date’’* PHASE:XX BOOT:vX.Y
where V = software version,
RR = software release,
CODE = software ID
dates format: dd/mm/yy.

CODE	product and software
9148	DMAC17 standard
A148	DMAC17 box version
B148	DMAC17 clock & dir
K148	DMAC17 CANopen
F138	DMAC23 standard
J138	DMAC23 clock & dir
K138	DMAC23 CANopen
H142	DMAC34 standard
J142	DMAC34 clock & dir
K142	DMAC34 CANopen

Example: `04 REQUEST_VERSION`
sample module answer:
04EV v1.7 H142 "MIDI-INGENIERIE_DMAC34-1_H142-10145_25/01/05_12/04/06" PHASE:6A BOOT:v1.1
Module is a DMAC34-1
Serial number is H142-10145
Manufacturing date: 21/01/05
Revision date: 12/04/06.
Firmware: H142 (DMAC43 standard)
Version/release: 1.7

4.20. SET_ADDRESS

Mnemonic	SAD parameter
Parameter:	New module address [0 ; 63] Factory default value = 0.
Description:	Modifies module address. Global commands are executed by all the modules but only the module at address 0 can send an answer. **Module address is stored in non-volatile memory.**
Note:	Two modules cannot have the same address (conflict). It is the responsibility of the user to handle correct module addressing. SET_ADDRESS command should not be sent in a global way (without specifying targeted module). WARNING: Factory default address is 0 for all modules. One should set address of a module BEFORE connecting it to a bus.

Example:	`04SET_ADDRESS 3`
	module 4 becomes module 3
	
	`04SET_ADDRESS 0`
	module 4 becomes module 0. From now on, it answers global commands.
	
	`SET_ADDRESS 6`
	Warning: global command! All modules are set to address 6. Conflict!

4.21. SET_BAUDRATE

Mnemonic	SBA parameter
Parameter:	defines communication speed in bauds (bits per seconds) 115200, 38400, 19200, 9600 (all other values will not be executed) Factory default value = 38400.
Description:	Specifies communication speed between host (PC or PLC) and the modules. **Baudrate is stored in non-volatile memory.**
Note:	Baudrate should be the same for all modules. `SET_BAUDRATE` command should be sent in a global way (without specifying targeted module). New baudrate is applied immediately and does not require a reset.
Example:	`SET_BAUDRATE 38400` From now on, modules will communicate at 384000 bauds.

4.22. S_CURVE

Syntax:	[@]S_CURVE parameter
Mnemonic:	SCU
Parameter:	ON (S-curve ramp) or OFF (trapezoidal ramp)
Description:	Enable/Disable S-curve velocity ramp. Compared to trapezoidal velocity ramp, S-curve results in a smoother movement, minimizing acceleration and torque discontinuity. **This parameter is stored in non-volatile memory.**
Note:	To disable velocity ramps ("start-stop"), one should set `#ACCEL_TIME` and `#DECEL_TIME` to zero.
Example:	`05S_CURVE OFF` Module 5 will use trapezoidal ramps for all movement.

4.23. SOFT_ENDS

Syntax:	[@]SOFT_ENDS parameter
Mnemonic:	SEN
Parameter:	ON (enable end-stops) or OFF (disable end-stops)
Description:	Enable/Disable software "virtual" end-stop. `#POSITIVE_END` defines maximal position (CW direction). `#NEGATIVE_END` defines minimal position (CCW direction). Beyond these two positions, all movement is stopped. **This parameter is stored in non-volatile memory.**
Example:	`02SOFT_ENDS OFF` Disable software end-stops for module 02.

4.24. START_SEQ

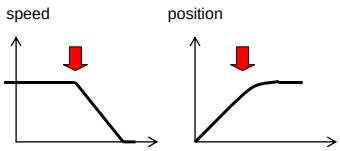
Syntax:	[@]START_SEQ [parameter]
Mnemonic:	SSE
Parameter:	Number of the first line to be executed (default: 1).
Description:	Execution of the sequence from specified line. If no parameter is given, execution starts at line 1.

Example:	`START_SEQ 15`
	All modules start sequencer from line 15.
	
	`02START_SEQ`
	Module 2 start sequencer from line 1.

4.25. STEP

Syntax:	[@]STEP [parameter]
Mnemonic:	STE
Parameter:	Sequencer line number to be executed.
Description:	Sequencer debug mode: If parameter > 0, specified sequencer line will be executed. If parameter is not specified, following sequencer line will be executed (#LINE is automatically incremented).
Example:	`04STEP 25` Module 4 executes sequencer line 25.

4.26. STOP

Syntax:	[@]STOP [parameter]
Mnemonic:	STO
Parameter:	None, MOUV or SEQ
Description:	Module stops, going from current speed to zero speed. Deceleration time is proportional to `#DECEL_TIME`.  Command STOP without any parameter stops both movement and sequence. "STOP MOUV" stops only the movement but keeps sequence running. "STOP SEQ" stops only the sequence but continues current movement. Note : In interpolated mode, STOP command is equivalent to HALT (immediate stop without deceleration) and interpolation FIFO is emptied.

Example:	`04STOP`
	Module 04 decelerates until complete stop. If a sequence is running, it is stopped.
	`03STOP MOUV`
	Module 03 decelerates until complete stop. If a sequence is running, it is not stopped.

4.27. SYNCHRO

Syntax:	SYNCHRO [parameter]
Mnemonic:	SYN
Parameter:	ON, OFF, TOP or INTERPOL
Description:	This command allows multi-axis synchronization. `SYNCHRO ON` enables synchro mode (useless for interpolation mode) `SYNCHRO OFF` disables synchro mode `SYNCHRO TOP` launch movement (except interpolation) `SYNCHRO INTERPOL` launch movement (interpolation) Note : Power is automatically switched on at the beginning of a movement. For better synchronization, it is recommended to switch it on before starting the movement (`POWER ON`).

Example:	1) standard movement (non interpolated)
	
	`SYNCHRO ON` ; enable synchro mode
	`00MOVE_SPEED 4000`
	`03MOVE_SPEED -12500`
	`SYNCHRO TOP` ; launch movements
	`SYNCHRO OFF` ; disable synchro mode
	
	2) interpolation (see MOVE_INTERPOL command at chapter 4.4)

4.28. WAIT

Syntax:	WAIT parameter
Mnemonic:	WAI
Parameters:	waiting time in milliseconds [-3600000 ; 3600000]
Description:	Execution of a sequence is suspended until delay is over. If parameter is 0 (e.g. "`WAIT 0`"), execution is suspended until current movement is completed. If parameter is negative (e.g. "`WAIT -1000`"), execution is suspended until current movement is completed, with a timeout specified by the parameter.

Example:	`WAIT 2000`
	Sequencer waits 2 seconds before going on to next line.
	
	`WAIT 0`
	Sequencer waits until movement is completed before going on to next line.
	
	`WAIT -2000`
	Sequencer waits for movement to be completed for a maximum of 2 seconds before going on to next line.

5. Internal variables

5.1. Syntax

Standardized syntax is as follows:
Writing a variable: [@]#VARIABLE[.BIT]:= [- ! H B]VALUE
Reading a variable: [@]READ [H B]#VARIABLE
Module answer: @MNEMONIC=VALUE

All DMAC variables are stored in a "32 bit signed" form.

5.1.1. Decimal form

By default, values are expressed in decimal form.
Character "+" before positive values is optional but always specified in modules answers.

Examples:
```
00#ACCEL_TIME:=123
00#V20:=-40
00READ #ACCEL_TIME → 00#ATI=+123
```

5.1.2. Hexadecimal form

A preceding "H" or "h" character shall be used to type and read hexadecimal values.
Negative values are represented in 4 bytes two's complement form (e.g. -10 = hFFFFFFF6).
Leading zeros are optional, but all values are read as 4 bytes (8 digits).

Examples:
```
00#ACCEL_TIME:=H100
00#V20:= HFFFFFFD8
00READ h#ACCEL_TIME → 00#ATI=h00000100
```

5.1.3. Binary form

A preceding "B" or "b" character shall be used to type and read binary values.
Negative values are represented in 4 bytes two's complement form.
Leading zeros are optional, but all values are read as 4 bytes (separated by a space).

Examples:
```
00#ACCEL_TIME:=B1100100
00READ b#ACCEL_TIME → 00#ATI=b00000000 00000000 00000000 01100100
```

5.1.4. Opposite and complement

A preceding "-" character shall be used to access the opposite of a variable.
A preceding "!" character shall be used to access the binary complement of a variable.
This notation is especially useful for writing sequences.

Examples:
```
00#V21:=-#V1
00#V23:=#V15 & !#STATUS
00MOVE_TO -#POSITION
```

5.1.5. Bit form

To read or write one bit of a variable, the name of she variable should be followed by a point ant the number of the bit (1 being the LSB, 32 being the MSB).
Bit value is then considered as a variable that can have value 1 or 0.
Bit form can be used in a command or a test.

Examples:

```
00#V10.3:=1                          ;sets third bit of #V10 to 1
00#V10.12:=0                         ;sets 12th bit of #V10 to 0
00READ #STATUS.5 → 00#STA.5=1        ;reads 5th bit of #STATUS
00#V12.1:=#STATUS.12 & #ERROR.2      ;logical AND between two bits
00IF #V1.24 = 1 JUMP 3               ;test if bit 24 of #V1 is set
```

5.1.6. Operations on variables

Several operations can be performed using variables:
[@]#VARIABLE:=OPERAND1 OPERATOR OPERAND2

There shall be at least one space character before and after the operator.
An operands can be a variable or a constant (decimal, hexadecimal, binary or bit form) .

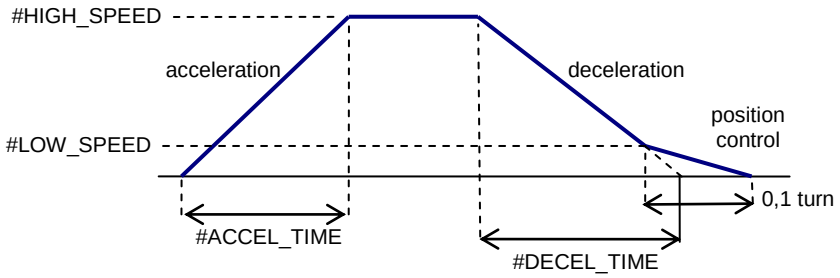
The operand must be one of the following::

operator	operation	example
+	addition	`#V1:=#POSITION + 12000`
-	subtraction	`#LOW_SPEED:=#HIGH_SPEED – #V3`
*	multiplication	`#V2:=3 * -#SUPPLY_VOLTAGE`
/	integer division	`#M3:=#POSITION / 10000`
&	bitwise AND	`#V2:=#STATUS & H00000200`
|	bitwise OR	`#M1:=!#V4 | H00000240`
>	test superior	`#V1:=#POSITION > 123`
<	test inferior	`#V1:=#POSITION < 123`
>=	test superior or equal	`#V1:=#POSITION >= 123`
<=	test inferior or equal	`#V1:=#POSITION <= 123`
!=	test different	`#V2:=#POSITION != 0`

Result of test operations is 1 if the test is TRUE and 0 if the test is FALSE. Test operations are to be used within sequences for conditional jumps (IF…JUMP).

5.1. #ACCEL_TIME, #DECEL_TIME

Mnemonic	#ATI, #DTI
Parameter:	acceleration and deceleration time, expressed in milliseconds. [0 ; 12000]
Description:	#ACCEL_TIME: time from standstill to #HIGH_SPEED #DECEL_TIME: time from #HIGH_SPEED to standstill.



This parameter is stored in non-volatile memory.

Note: Variable `#ACCEL_TIME` represents the time to go from standstill (stopped) to target velocity `#HIGH_SPEED`.
Variable `#ACCEL_TIME` represents the time to go from target velocity `#HIGH_SPEED` to standstill.

If current velocity or target velocity are different from standstill or #HIGH_SPEED, acceleration and deceleration time is automatically computed proportionnaly to speed difference, so as to obtain constant accelerations.

$$\text{acceleration time} = \#\text{ACCEL\_TIME} \times \frac{|\text{Target velocity - Current velocity}|}{\#HIGH\_SPEED}$$

$$\text{deceleration time} = \#\text{DECEL\_TIME} \times \frac{|\text{Target velocity - Current speed}|}{\#HIGH\_SPEED}$$

Example:
`#ACCEL_TIME:=1000` (1s)
`#HIGH_SPEED:=60000` (600tr/min)
`MOVE_SPEED 30000` → computed acceleration time: 0.5s

Example: `00#ACCEL_TIME:=1000`
Module will go from standstill to `#HIGH_SPEED` in one second.

5.2. #CAPTURE (read only)

Mnemonic	#CAP
Parameter:	Value of `#POSITION` at last IN5 edge.
Description:	At each rising edge (0→1) of IN5 digital input, current position is stored in `#CAPTURE` variable.
Example:	`03READ #CAPTURE` → `03#CAP=27895` Module 3 was at position 27895 at IN5 rising edge.

5.3. #CPU_TEMPERATURE (read only)

Mnemonic	#CTE
Parameter:	Internal microcontroller temperature (units = 0,1°C)
Example:	`02READ #CPU_TEMPERATURE` → `02#CTE=520` CPU has a temperature of 52°C on module 2.

5.4. #DRIVER_TEMPERATURE, #MOTOR_TEMPERATURE (read only, DMAC34)

Mnemonic	#DTE (DMAC34 only) #MTE (DMAC34 only)
Parameter:	#DTE indicates DMAC34 power stage temperature (unit = 0,1°C) #MTE indicates DMAC34 motor temperature (unit = 0,1°C)
Example:	`02READ #MOTOR_TEMPERATURE` → `02#MTE=650` Motor on DMAC34 at address 2 is 65°C hot.

5.5. #ERROR

Mnemonic: #ERR

Description: DMAC error register:

MSB	bit 32	
	…	
	bit 12	Command or Variable is undefined (syntax)
	bit 11	
	bit 10	
	bit 9	
	bit 8	Computational error (divide by zero…)
	bit 7	Limit (parameter beyond limit)
	bit 6	
	bit 5	Overvoltage
	bit 4	Undervoltage
	bit 3	Short-circuit
	bit 2	Thermal disjunction
LSB	bit 1	

Error acknowledgement is done by writing 0 in `#ERROR`

Example:
```
01READ b#ERROR
→ 01#ERR=b00000000 00000000 00000000 00010000
```

Reading error register of module 1 (binary form) → overvoltage detected.

5.6. #HIGH_SPEED

Mnemonic	#HSP
Parameter:	Target velocity (expressed in 0.01RPM) [0 ; 400000]
Description:	Target velocity for position mode movement (`MOVE_TO` et `MOVE_ON`) and maximal velocity for speed mode (`MOVE_SPEED`). **This parameter is stored in non-volatile memory.**
Note:	If movement length is too short, target velocity may not be reached.
Example:	`04#HIGH_SPEED:=20000` Next movements will be done at 200RPM for module 4.

5.7. #INPUT (read only)

Mnemonic #INP

Parameter: Value representing digital inputs (binary active if 1, inactive if 0)

Description: Digital inputs from IN1 (LSB) to IN6 or IN10 (MSB, depending on model).
If polarity is inverted (command `INVERSE_POLARITY` represented by bit `#STATUS.13`), the logical state of `#INPUT` is inverted versus the physical state.

Example: 1) DMAC23 at address 01 (with `INVERSE_POLARITY OFF`)

`01READ #INPUT` → `01#INP=19`

19 (decimal) =	0	0	0	1	0	0	1	1	(binary)
			OFF	ON	OFF	OFF	ON	ON	
	----	----	IN6	IN5	IN4	IN3	IN2	IN1	

2) DMAC34 at address 05 (with `INVERSE_POLARITY OFF`)

`05READ #INPUT` → `05#INP=19`
`05READ h#INPUT` → `05#INP=h13`
`05READ b#INPUT` → `05#INP=b00000000000000000000000000010011`

19 (decimal) =	0	0	0	0	0	1	0	0	1	1	(binary)
	OFF	OFF	OFF	OFF	OFF	ON	OFF	OFF	ON	ON	
	IN10	IN9	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	

5.8. #INPUT_ANALOG, #INPUT_A1, #INPUT_A2 (read only)

Mnemonic	#IAN (DMAC17 and DMAC23) #IA1, #IA2 (DMAC34)
Parameter:	Analog input differential voltage in mV.

Example:	`02READ #INPUT_ANALOG` → `02#IAN=-3200`
	Module at address 02 has a voltage of -3.2V on its analog input.
	
	`04READ #INPUT_A2` → `04#IA2=1500`
	DMAC34 at address 04 has a voltage of 1.5V on its second analog input.

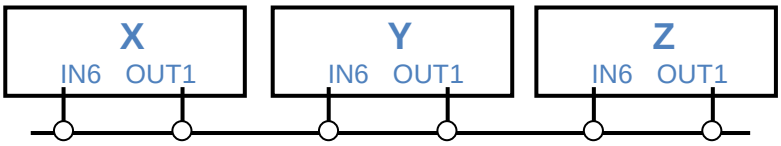
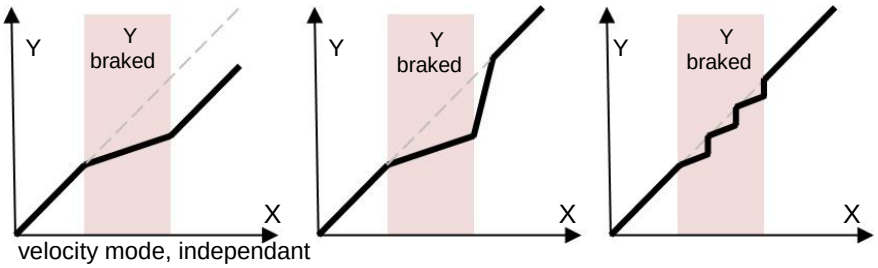
5.9. #INTERPOL_COUNT (read only)

Mnemonic	#ICO
Parameter:	Number of segments in FIFO memory [0 ; #INTERPOL_FIFOSIZE]
Description:	When FIFO is empty, #INTERPOL_COUNT=0. Movement is stopped after execution of the last segment. (see command MOVE_INTERPOL for detailed information on interpolation mode)
Note:	Available space in FIFO is given by the difference: `#INTERPOL_FIFOSIZE - #INTERPOL_COUNT`
Example:	`02READ #INTERPOL_COUNT` → `02#ICO=23` => 23 segments are currently stored in axis 02.

5.10. #INTERPOL_FIFOSIZE

Mnemonic	#IFI
Parameter:	Size of the FIFO memory, expressed in segments [1 ; 64]
Description:	(see command MOVE_INTERPOL for detailed information on interpolation mode)
Note:	Setting a high value for #INTERPOL_FIFOSIZE can be less demanding for real-time capabilities of host communication. DMAC tolerates a certain latency between MOVE_INTERPOL messages. Setting a low value will be useful if a low response time is needed (for example when trajectory is computed in real-time).
Example:	`#INTERPOL_FIFOSIZE :=64` ; max value for all axis

5.11. #INTERPOL_MODE

Syntaxe	[@]#INTERPOL_MODE:=parameter
Mnemonic:	#IMO
Parameter:	0 (SimpleSYNC) or -1 (UltraSYNC)
Description:	This variable can choose between the two synchronization types for interpolation mode. **SimpleSYNC mode** (`#INTERPOL_MODE:=0`) Multi-axis synchronization is done by simultaneous start of the movement on all the modules (global command `SYNCHRO INTERPOL`). The duration of each segment is then assured by a precision internal timer. **UltraSYNC mode** (`#INTERPOL_MODE:=-1`) Multi-axis synchronization is done by chaining digital IOs IN6 and OUT1 for all the modules. If one of the axis goes slower than others (mechanical friction…) all the other axis are automatically informed and wait for the segment to be finished on all modules. 
Example:	Example on a 2 axis configuration X-Y:  velocity mode, independant axis (non interpolated) — SimpleSYNC — UltraSYNC

5.12. #INTERPOL_TIME

Mnemonic	#ITI
Parameter:	Duration of one interpolation segment in ms [2 ;138]
Description:	New length will be applied on all next `MOVE_INTERPOL` commands. (see command MOVE_INTERPOL for detailed information on interpolation mode) In most cases, all axis should use the same value for #INTERPOL_TIME.
Note:	To keep the trajectory synchronized, #INTERPOL_TIME should be modified at the same time on all axis.
Example:	`#INTERPOL_TIME:=50` ; all future segments will last 50ms

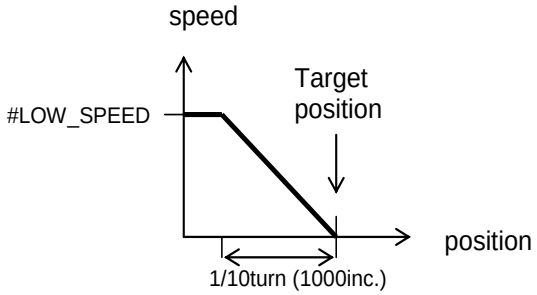
5.13. #LINE_DELAY

Mnemonic	#LDE
Parameter:	RS485 turn-around delay in microseconds [100 ; 3000] Factory default value = 3000.
Description:	RS485 delay between the end of a command from host (PC or PLC) and the beginning of the answer from the DMAC. **This parameter is stored in non-volatile memory.**
Note:	#LINE_DELAY should be set to the same value for all the axis.
Example:	`#LINE_DELAY:=1000` Module will wait for 1ms after having received a command before sending the answer.

5.14. #LINE

Mnemonic	#LIN
Parameter:	Sequencer line currently executed [0 ; 500]
Note:	When sequencer is not started, `#LINE`=0. Resetting this variable (`#LINE:=0`) stops the sequencer.
Example:	`READ #LINE` → `00#LIN:=182` Module at address 0 is executing line 182 of the sequence.

5.15. #LOW_SPEED

Mnemonic	#LSP
Parameter:	Approach speed in 100th rev./min [0 ; 400000]
Description:	#LOW_SPEED is used at the end of position mode movement (`MOVE_TO` and `MOVE_ON`). Velocity profile during position control loop is defined by `#LOW_SPEED`:  **This parameter is stored in non-volatile memory.**
Note:	This variable permits fine tuning of the response time and stability of the position closed-loop control. Response time is shorter as #LOW_SPEED is set to a high value. On the other hand, if #LOW_SPEED is set too high, an oscillation can appear if load inertia is important, with little friction.
Example:	`04#LOW_SPEED:=2000` In position mode, Position control will go from 20RPM to standstill at the end of any movement.

5.16. #M1 to #M8

Mnemonic	#M1 to #M8
Parameter:	Memorized user variable. 4 bytes signed format [-2 147 483 648 ; + 2 147 483 647]
Description:	User variables: can be used to store any kind of value. `#M1` to `#M8` content is stored in non-volatile memory and restored at power-on, unlike `#V1` to `#V32` which content is reset.
Example:	`#M2:=H100` Store hexadecimal value 0x100 in variable `#M2`.

5.17. #NEGATIVE_END

Mnemonic	#NEN
Parameter:	Virtual negative end-stop (10^{-4}rev.) [-2 147 483 648 ; + 2 147 483 647]
Description:	All movement going beyond this position is stopped. #STATUS.20 bit is set and only CW movement is enabled. **This parameter is stored in non-volatile memory.**
Note:	`#NEGATIVE_END` value is not modified when `#POSITION` is modified.
Example:	`03#NEGATIVE_END:=-80000` Module at address 3 can only go 8 rev. CCW away from home position.

5.18. #ON_RESET

Mnemonic	#ORE
Parameter:	Number of sequencer line to be executed on reset [0 ; 500]
Description:	Sequencer will automatically be launched, starting from specified line, on reset. To disable auto-start of the sequencer, set value to zero: `#ON_RESET:=0.` **This parameter is stored in non-volatile memory.**
Example:	`#ON_RESET:=1` Module executes sequencer, starting from line 1 at reset.

5.19. #OUTPUT

Mnemonic	#OUT
Description:	Digital outputs. If the polarity is inverted (command `INVERSE_POLARITY` represented by bit `#STATUS.12`),the logical state of `#OUTPUT` is inverted versus output physical state.
Note:	All 32 bits of variable `#OUTPUT` can be written, but only LSB correspons to physical outputs (for instance `#OUTPUT.1` to `#OUTPUT.4` for modules with 4 outputs, or `#OUTPUT.1` to `#OUTPUT.8` for modules with 8 outputs). OUT1 and OUT2 can be used for alternate functions Busy and Default according to `#OUTPUT_CONFIG` variable. The physical state of the output is then determined by module status instead of `#OUTPUT` variable.
Example:	`00#OUTPUT:=4` Activates OUT3, Deactivates all others 4 (decimal) = 0 0 0 0 0 1 0 0 (binary) OUT8 OUT7 OUT6 OUT5 OUT4 OUT3 OUT2 OUT1

5.20. #OUTPUT_A1, #OUTPUT_A2 (DMAC34 only)

Mnemonic	#OA1, #OA2
Parameter:	Value of the voltage to be applied on analog outputs 1 or 2, expressed in mV [0 ; + 10 000].
Note:	This feature is available with DMAC34 only. See description of analog outputs for details. Can be used within a sequence.
Example:	`00#OUTPUT_A1:=4520` Sets 4.52V on analog output 1.

5.21. #OUTPUT_CONFIG

Mnemonic	#OCO
Description:	Digital outputs multiplexing with alternate functions. Outputs 1 et 2 can be used as GPIO or as alternate functions: OUT1 can represent BUSY signal OUT2 can represent FAULT signal If `#OUTPUT_CONFIG` bit is 0, output is used as a GPIO. Output state is determined by corresponding `#OUTPUT` bit. If `#OUTPUT_CONFIG` bit is 1, output is used as an alternate function. Output state is determined by the status of the module. **This parameter is stored in non-volatile memory.**
Note:	Default factory value for `#OUTPUT_CONFIG` is 3 (OUT1 and OUT2 represent BUSY and FAULT signals).
Example:	`00#OUTPUT_CONFIG:=0` All outputs are used as GPIOs `00#OUTPUT_CONFIG:=1` OUT1 = BUSY, OUT2 = GPIO `00#OUTPUT_CONFIG:=2` OUT1 = GPIO, OUT2 = FAULT `00#OUTPUT_CONFIG:=3` OUT1 = BUSY, OUT2 = FAULT

5.22. #POSITION

Mnemonic	#POS
Parameter:	New position (10^{-4}rev.) [-2 147 483 648 ; + 2 147 483 647]
Description:	Force module position. Can be used to set home position (`#POSITION:=0`).
Note:	**Module position is stored in non-volatile memory.** **The value may not be correct if the motor has moved when not powered.** Warning: do not set a position outside end-stops range (defined by #NEGATIVE_END and #POSITIVE_END) if they are enabled.
Example:	`04#POSITION:=0` Set new "home" reference position for module at address 4.

5.23. #POSITIVE_END

Mnemonic	#PEN
Parameter:	Virtual positive end-stop (10^{-4}rev.) [-2 147 483 648 ; + 2 147 483 647]
Description:	All movement going beyond this position is stopped. #STATUS.19 bit is set and only CCW movement is enabled. **This parameter is stored in non-volatile memory.**
Note:	`#POSITIVE_END` value is not modified when `#POSITION` is modified
Example:	`#POSITIVE_END:=100000` Module can only go 100 rev. CW away from home position.

5.24. #SPEED, #PROFILE_SPEED (read only)

Mnemonic	#SPE, #PSP
Parameter:	Axis velocity in 100th of RPM. Signed
Description:	Variable `#SPEED` is the current motor speed (measured). Variable `#PROFILE_SPEED` is the computed speed of the module.
Note:	The value of `#SPEED` is determined by sampling module position. Instantaneous value can then be slightly different from real speed.
Example:	`04READ #SPEED` → `04#SPE=-20000` Reading speed of module 4. Module 4 rotates at 200RPM in CCW direction.

5.25. #STATUS (read only)

Mnemonic #STA

Description: DMAC status word:

MSB	bit 32	Movement stopped abnormally
	bit 31	Error (check #ERROR for details)
	bit 30	Reference mode
	bit 29	Busy
	bit 28	
	bit 27	Position control
	bit 26	Movement
	bit 25	Power ON (1) or OFF (0)
	bit 24	
	bit 23	Synchro mode
	bit 22	
	bit 21	Brake active (1) or inactive (0, see BRAKE command)
	bit 20	Soft negative end-stop
	bit 19	Soft positive end-stop
	bit 18	Hard negative end-stop
	bit 17	Hard positive end-stop
	bit 16	Sequencer Edition mode
	bit 15	Sequencer run
	bit 14	Output polarity standard (0) or inverted (1)
	bit 13	Input polarity standard (0) or inverted (1)
	bit 12	Velocity profile « S » (1) or trapezoidal (0)
	bit 11	
	bit 10	
	bit 9	
	bit 8	Inverted polarity of hard end-stops inputs
	bit 7	Enable soft end-stops
	bit 6	Enable hardware negative end-stops
	bit 5	Enable hardware positive end-stops
	bit 4	Optimized current mode
	bit 3	
	bit 2	
LSB	bit 1	

Example: `03READ h#STATUS` → `03#STA=h13000800`
Reading status in hexadecimal form.

5.26. #SUPPLY_VOLTAGE (read only)

Mnemonic #SVO

Parameter: Measure of supply voltage in mV

Example: `02READ #SUPPLY_VOLTAGE` → `02#SVO=32000`
Module 2 supply voltage is 32V.

5.27. #TIMER_1 to #TIMER_3

Mnemonic	#T1 to #T3
Parameter:	Tempo in milliseconds [0 ; + 2 147 483 647]
Description:	Variables `#TIMER_1` to `#TIMER_3` are to be used within a sequence to insert a delay. Starting from a value defined by the user, those three variables are decremented every millisecond down to zero. They can be read (`READ #TIMER_1`) or tested (`IF #TIMER_1=0 JUMP…`) to know if the delay is over.
Example:	`#TIMER_3:=2000` Variable `#TIMER_3` is loaded with value 2000. It will be decremented every millisecond for the next two seconds.

5.28. #TORQUE_RATIO

Mnemonic	#TRA
Parameter:	Percentage of available torque. [0 ; 100]
Description:	Set this variable to a smaller value when all the torque is not necessary. Power consumption and thermal losses will then be reduced. **This parameter is stored in non-volatile memory.**
Note:	Max speed and torque capabilities of the motor can only be attained at `#TORQUE_RATIO:=100`.
Example:	`00#TORQUE_RATIO:=70` Module 0 uses only 70% of the nominal torque.

5.29. #V1 to #V32

Mnemonic:	#V1 to #V32
Parameter:	4 bytes signed value [-2 147 483 648 ; + 2 147 483 647]
Description:	Variables `#V1` to `#V32` can be used to store values, parameter or math results. They can be used within a sequence. They are initialized to zero at reset or power-up.
Example:	`00#V13:=1234` Stores value 1234 in variable `#V13` of module 0.

6. Sequencer

6.1. Features

DMAC can store and then execute command lines.
Each line is represented by its "line number" (1 to 500).
When sequencer is at line zero (`#LINE=0`), it is not running.
When sequencer is running (command "`START_SEQ n`" where `n` is the first line to be executed), the module executes one line every millisecond and goes on to next line (automatically incrementing `#LINE`).

6.2. Lines storage

To store sequence lines in non-volatile memory:

1. Enable "Edit mode" using command `OPEN_SEQ`.
2. Type commands just as if they were to be executed in "Live mode". The line number is automatically incremented or can be forced by preceding the command with " `:n` " where n is the line number.
   (Example: "`00:120 MOVE_TO 15000`")
3. Disable "Edit mode" with command `CLOSE_SEQ`

Example: Storage of a sequence that makes one rev every two seconds:

```
OPEN_SEQ          ;enable Edit mode
MOVE_ON 10000     ;1 rev CW
WAIT 2000         ;delay
JUMP 1            ;go back to line 1
CLOSE_SEQ         ;disable Edit mode
```

The following command lines will be stored:
Line 1 : `MOVE_ON 10000`
Line 2 : `WAIT 2000`
Line 3 : `JUMP 1`

6.3. Execution of the sequencer

Use `START_SEQ` command to launch sequencer execution. First line number can be specified as an argument if needed.

To stop sequencer execution, use command `STOP` (or `STOP SEQ` to keep motor movement).
Line 0 corresponds to sequencer stop. When the module is at line 0, it stays there until it receives a `START_SEQ` command.

The sequencer can be configured to start execution at module reset if `#ON_RESET` variable contains a value.

Example: To start sequencer from line 12 on reset, type " `#ON_RESET:=12` ".
To disable this feature, type `#ON_RESET:=0`

6.4. Sample sequences

6.4.1. Example 1

Digital output 4 is ON if supply voltage is in the range 15V - 20V.

```
OPEN_SEQ                              ;enter Edit mode
IF #SUPPLY_VOLTAGE < 15000 JUMP 5     ;output OFF if V < 15Volts
IF #SUPPLY_VOLTAGE > 20000 JUMP 5     ;output OFF if V > 20Volts
#OUTPUT.4:=1                          ;output ON otherwise
JUMP 1                                ;loop on line 1
#OUTPUT.4:=0                          ;output OFF
JUMP 1                                ;loop on line 1
CLOSE_SEQ                             ;quit Edit mode
START_SEQ 1                           ;Sequencer start
```

6.4.2. Example 2

Speed control loop: The DMAC velocity is determined by analog input with gain and offset.
(this example use mnemonics instead of full-size commands)

```
OSE                                   ;enter Edit mode (OPEN_SEQ)
:50 #V1:=#IAN - 15000                 ;line 50: #INPUT_ANALOG - 15000 in V1
#V1:=#V1 * 5                          ;gain of 5
MSP #V1                               ;movement (MOVE_SPEED)
JRE -3                                ;loop on 50 (JUMP_REL)
CSE                                   ;quit Edit mode (CLOSE_SEQ)
SSE 50                                ;Sequencer start (START_SEQ)
```

6.4.3. Example 3

Digital input 2 triggers 1 rev clockwise and digital input 3 triggers 1 rev counter-clockwise.

```
OPEN_SEQ                        ;enter Edit mode
IF #INPUT.2 = 1 JUMP_REL +3     ;test input 2
IF #INPUT.3 = 1 JUMP_REL +4     ;test input 3
JUMP 1                          ;loop on line 1
MOVE_ON 10000                   ;1 rev CW
JUMP_REL 2                      ;jump to "wait for the end of the movement"
MOVE_ON -10000                  ;1 rev CCW
WAIT 0                          ;wait for the end of the movement
JUMP 1                          ;loop on line 1
CLOSE_SEQ                       ;quit Edit mode
START_SEQ 1                     ;Sequencer start
```

6.4.4. Example 4

Call subroutine to reset position if input 3 is OFF or if supply voltage is above 30V.

```
OPEN_SEQ                                ;enter Edit mode at line 1
:150 IF #INPUT.3 = 0 CALL 160           ;Line 150: test input 3
IF #SUPPLY_VOLTAGE > 30000 CALL 160     ;test supply voltage
JUMP 0                                  ;stop sequencer

:160 #POSITION:=0                       ;Line 160: reset position
RETURN                                  ;back to CALL
CLOSE_SEQ                               ;quit Edit mode
START_SEQ 150                           ;Sequencer start
```

7. ANNEX

7.1. Commands abstract

Command	Mnemo.	Parameter	Live	Sequence	Factory value (MRE ALL)
BRAKE (*)	BRA	ON / OFF	◊	◊	ON
CALL	CAL	seq line		◊	
CLOSE_SEQ	CSE	-	◊		
HALT	HAL	- / SEQ / MOUV	◊	◊	
HARD_ENDS	HEN	OFF / ALL / POS / NEG	◊	◊	OFF
IF	IF	test		◊	
INVERSE_POLARITY	IPO	OFF / ALL / IN / OUT	◊	◊	OFF
JUMP	JUM	seq line		◊	
JUMP_REL	JRE	seq line		◊	
MODULE_RESET	MRE	- / ALL	◊		
MOVE_INTERPOL	MIN	relative position	◊		
MOVE_ON	MON	relative position	◊	◊	
MOVE_SPEED	MSP	target speed	◊	◊	
MOVE_TO	MTO	absolute position	◊	◊	
OPEN_SEQ	OSE	-	◊		
OPTIMIZED_CURRENT	OCU	ON / OFF	◊	◊	OFF
POWER	POW	ON / OFF / SC	◊	◊	OFF
READ	REA	variable	◊		
READ_SEQ	RSE	seq line	◊		
REFERENCE	REF	ON / OFF	◊	◊	OFF
RETURN	RET	-		◊	
REQUEST_VERSION	RV	-	◊		
SET_ADDRESS	SAD	address	◊		0
SET_BAUDRATE	SBA	baudrate	◊		38400
S_CURVE	SCU	ON / OFF	◊	◊	OFF
SOFT_ENDS	SEN	ON / OFF	◊	◊	OFF
START_SEQ	SSE	seq line	◊		
STEP	STE	seq line	◊		
STOP	STO	- / SEQ / MOUV	◊	◊	
SYNCHRO	SYN	ON / OFF /TOP / INTERPOL	◊		OFF
WAIT	WAI	delay		◊	

(*) DMAC34 only

7.2. Variable abstract

Variable	Mnemo.	Stored	Read only	Factory value (MRE ALL)
#ACCEL_TIME	#ATI	◊		1 000
#CAPTURE	#CAP		◊	
#CPU_TEMPERATURE	#CTE		◊	
#DECEL_TIME	#DTI	◊		1 000
#DRIVER_TEMPERATURE (*)	#DTE		◊	
#ERROR	#ERR			
#HIGH_SPEED	#HSP	◊		60 000
#INPUT	#INP		◊	
#INPUT_ANALOG	#IAN		◊	
#INPUT_A1, #INPUT_A2 (*)	#IA1, #IA2		◊	
#INTERPOL_COUNT	#ICO		◊	
#INTERPOL_FIFOSIZE	#IFI			64
#INTERPOL_MODE	#IMO			0
#INTERPOL_TIME	#ITI			100
#LINE_DELAY	#LDE	◊		3 000
#LINE	#LIN			0
#LOW_SPEED	#LSP	◊		6 000
#M1 to #M8	#M1 to #M8	◊		0
#MOTOR_TEMPERATURE (*)	#MTE		◊	
#NEGATIVE_END	#NEN	◊		-100 000
#ON_RESET	#ORE	◊		0
#OUTPUT	#OUT			0
#OUTPUT_A1, #OUTPUT_A2 (*)	#OA1, #OA2			0
#OUTPUT_CONFIG	#OCO	◊		3
#POSITION	#POS	◊		0
#POSITIVE_END	#PEN	◊		+100 000
#PROFILE_SPEED	#PSP		◊	
#SPEED	#SPE		◊	
#STATUS	#STA		◊	
#SUPPLY_VOLTAGE	#SVO		◊	
#TIMER_1 to #TIMER_3	#T1 to #T3			0
#TORQUE_RATIO	#TRA	◊		50
#V1 to #V32	#V1 to #V32			0

(*) DMAC34 only

7.3. Related Documents

Check midi-ingenierie.com website for the following docmuents.

- Note d'application – Liaison calculateur : protocoles et syntaxe
- Note d'application – DMAC Horloge & Sens
- Résumé des commandes DMAC et microMAC (français)
  Commands Abstract – DMAC and microMAC (anglais)
- Notice d'utilisation du bornier DMAC / microMAC
- Note d'application DMAC CAN