www.conairnet.com

USERGUIDE

SYSTEM CONFIGURATION for S900-II robots Software Version 2.1

⚠ **WARNING - Reliance on this Manual Could Result in Severe Bodily Injury or Death!**

This manual is out-of-date and is provided only for its technical information, data and capacities. Portions of this manual detailing procedures or precautions in the operation, inspection, maintenance and repair of the product forming the subject matter of this manual may be inadequate, inaccurate, and/or incomplete and cannot be used, followed, or relied upon. Contact Conair at info@conairgroup.com or 1-800-654-6661 for more current information, warnings, and materials about more recent product manuals containing warnings, information, precautions, and procedures that may be more adequate than those contained in this out-of-date manual.

Logo definitions :

	Warning, risk		Document evolutions
	Sepro robotique innovations		Handy hints
	What to do ?		Example
			Software innovation

– CONTENTS –

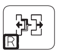
I – MEMORY

I – 1. Memory read function

Access : see next page

The address of the area in which reading is to begin is given in hexadecimal (0 to F) using the numerical keypad and the first row of alphanumerical keys of the keyboard.

Certain areas are directly accessible from the keyboard :

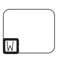
 : beginning of the PRG editing area (0 x 006 430).

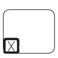
 : beginning of the PLC editing area (0 x 009 430).

 : beginning of the program storage in RAM area (0 x 00B 300).

 : beginning of the MODULE where the programs are stored (0 x 800 000).

 : robot serial number in RAM.

 : beginning of parameters in RAM.

 : beginning of the faults 200 to 204 message table in RAM.

For example : to access the beginning of the program storage area, the procedure is as follows :

[EXPLORER] –> [M_Read] –> [Address] –> 

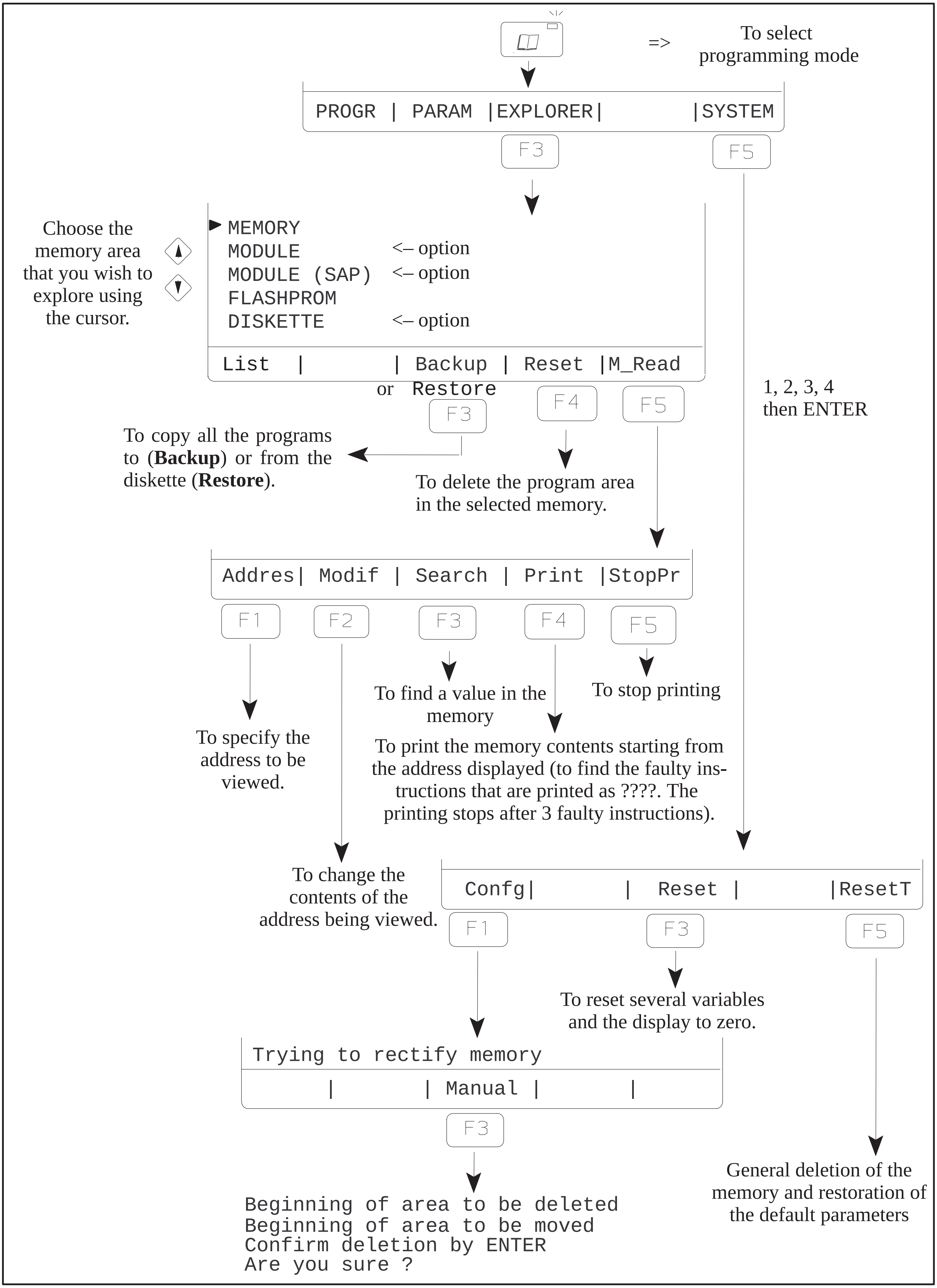
* **The keys** :

- ▶ [+] or [–] to change addresses 2 by 2.
- ▶ [↑] or [↓] to change addresses 10 by 10 (hexadecimal).
- ▶ [PG DN] or [PG UP] to change addresses 100 by 100 (hexadecimal).

Note : To access the modification function, you must enter a password that stays valid as long as you are in the “M_Read” procedure. Certain critical system areas can only be read and all requests to modify them will be rejected.

By default, the value given after requesting a modification is 0 x FFFF (useful for deleting words in the memory).

As for the other functions, use the EXIT key to abort a request or exit the procedure.


=>
To select programming mode
PROGR | PARAM |EXPLORER| |SYSTEM
F3
F5
Choose the memory area that you wish to explore using the cursor.
MEMORY
MODULE <– option
MODULE (SAP) <– option
FLASHPROM
DISKETTE <– option
List | | Backup | Reset |M_Read
or Restore
F3
F4
F5
1, 2, 3, 4 then ENTER
To copy all the programs to (Backup) or from the diskette (Restore).
To delete the program area in the selected memory.
Addres| Modif | Search | Print |StopPr
F1
F2
F3
F4
F5
To find a value in the memory
To stop printing
To specify the address to be viewed.
To print the memory contents starting from the address displayed (to find the faulty instructions that are printed as ????. The printing stops after 3 faulty instructions).
To change the contents of the address being viewed.
Confg| | Reset | |ResetT
F1
F3
F5
To reset several variables and the display to zero.
Trying to rectify memory
| | Manual | |
F3
General deletion of the memory and restoration of the default parameters
Beginning of area to be deleted
Beginning of area to be moved
Confirm deletion by ENTER
Are you sure ?

I – 2. Memory areas

I – 2. 1.Data saved in RAM (512 K x 8) 0 to 7 FFFF

Address in Hexadecimal	Contents
00000 027FF	Variables used by Philips (BOOT)
02800 0A4FF	“Fixed” SEPRO variables, see table below for details of the variables
0A500 0B2FF	SEPRO parameters in RAM
0B300 2A6FF	PRG storage area (128 K × 8)
2A700 37FFF	SEPRO variables / work tables
38000 57FFF	Temporary transfer area (128 K x 8)
58000 7FFFF	Piles and heaps used by the ERM kernel

02800	En Ordre = RAM contents correct indicator (GIRLAFRIDOU).
02810	Bit_U_S = System and user bits table.
02890	Bit_Tpo = PLC timer bits table.
028A0	Imag_S = Images of the 255 ON/OFF outputs.
029A0	Imag_E = Image of the 255 ON/OFF inputs.
02AA0	Word_U = User words table (16–bit WORD).
02AE0	Word_S = System words table (see Programming Level 2 manual for description).
02B20	Tpo_Aut = PLC timers table.
02B40	Compt = Counters table (standard and stacking).
04AA0	Pile_Def = Pile of historic faults.
04BC0	Comptime = Times basic counter.
04BC4	Dir_RAM = PRG / PLC directory in editing area.
04C04	Dir_PP = PRG directory in save area.
05254	Dir_PLC = PLC directory in save area.
05710	Mod_PP = PRG directory in the module.
05D60	Mod_PLC = PLC directory in the module.
0621C	Tab_temps = Robot times table.
06230	WWord_U = Double words table (32 bits).
06430	Ram_PP = PRG editing area.
09430	Ram_PLC = PLC editing area.

I – 2. 2.Program addresses in the memory

The PRG and PLC programs are stored in the RAM memory, starting from the address 0xB300.

The maximum length of a PRG is 12286 bytes ; 4096 bytes for a PLC.

This area reserved for the permanent storage varies depending on the option 32 to 128 Kbytes.

So that it remains compatible with previous software versions, the RAM is formatted with 0xFFFF like an EEPROM. This formatting is carried out when the robot is first started up (for the 128 Kbytes) or when the memory is totally set to 0 [ResetT] (on the size provided for in the options)

The parameters are stored in FLASHPROM at the address 0xF10E0000. An image of this address is stored in RAM at the address 0xA500. The length of the parameters is fixed at 2800 bytes.

The “SAP message” file is stored in FLASHPROM at the address 0xF10E1300. Its length is fixed at 4798 bytes.

The programs, parameters and SAP messages are transferred via a temporary buffer of 12286 bytes at the address 0x38000. (This buffer can be extended to 128 Kbytes).

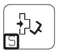
I – 2. 3.Data in Flashprom (1 M x 8) F10 00000 to F10 FFFFF

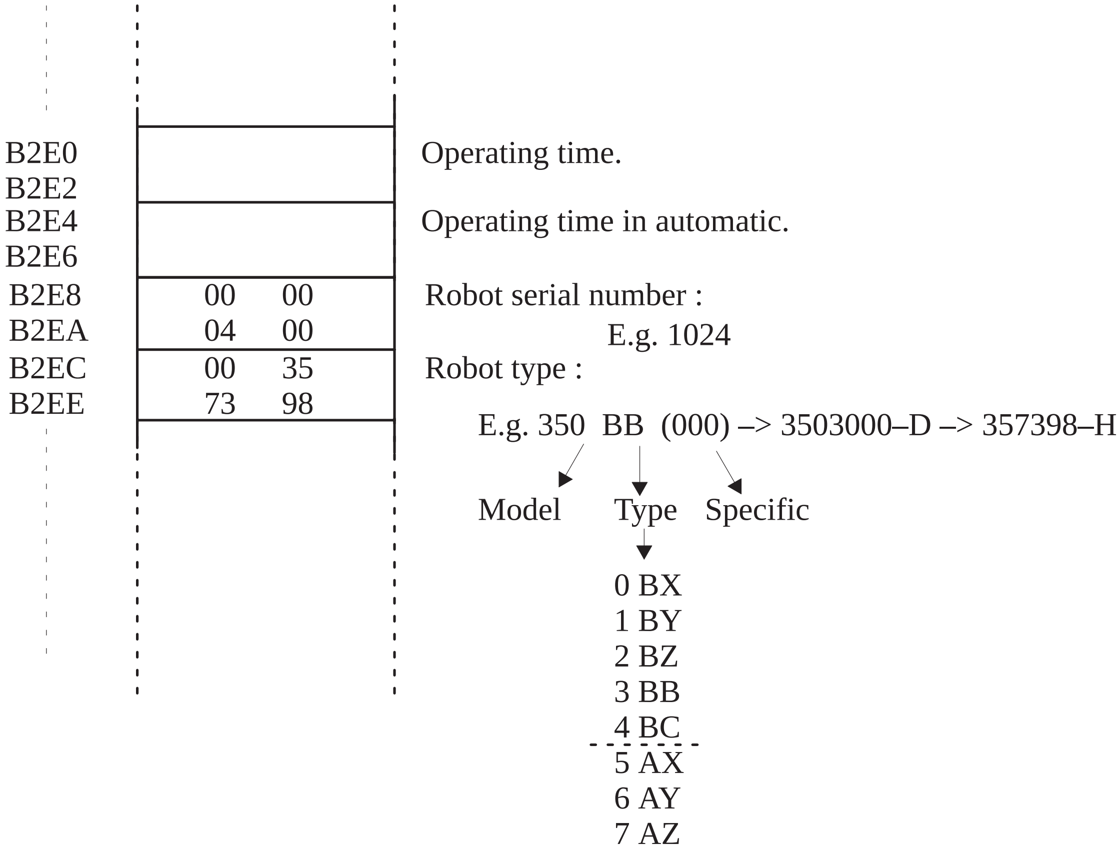
Block number	Address in Hexadecimal	Contents
1st block	F10 00000 F10 0FFFF	ERM kernel + SEPRO program
	F10 10000 F10 1FFFF	SEPRO code (1)
2nd block	F10 20000 F10 3FFFF	SEPRO code (2)
3rd block	F10 40000 F10 5FFFF	SEPRO code (3)
4th block	F10 60000 F10 7FFFF	SEPRO code (4)
5th block	F10 80000 F10 9FFFF	SEPRO code (5)
6th block	F10 A0000 F10 BFFFF	Reserved for extension of SEPRO code

Block number	Address in Hexadecimal	Contents
7th block Messages	F10 C0000 F10 CEBEF	Messages in language 1
	F10 CEBF0 F10 DD7DF	Messages in language 2
	F10 DD7E0 F10 DE7EF	Font robot 1
	F10 DE7F0 F10 DF7FF	Font robot 2
	F10 DF800 F10 DF9FF	Code converter table IMM 1
	F10 DFA00 F10 DFBFF	Code converter table IMM 2
	F10 DFC00 F10 DFDFF	Code converter table Printer 1
	F10 DFE00 F10 DFFFF	Code converter table Printer 2
8th block Parameters and SAP	F10 E0000 F10 E0DFF	SEPRO parameters
	F10 E1300 F10 E25FF	SAP messages
	F10 E2600 F10 F25FF	SAP and PLC programs (64Kb)

I – 3. Specific information

This is directly accessed using the Memory Read function followed by the request [Address] and a letter :

–  to access the memory area containing the serial number and the type of robot.



II – INSTRUCTION CODES

II – 1. Part programs

Type of instruction	Display	Codop (hexadecimal)	Examples
ACTION	ACT 00 –> 99 *	A000 [oper. 16 bits] ↓ Action number	A000000C = ACT12
OUTPUT	OUT 000 –> 255	A001 [oper. 16 bits] ↓ Output number	A0010050 = OUT 080
INPUT Normal	IN 000 –> 255	A002 [oper. 16 bits] ↓ Input number	A002000A = IN 010
INPUT Reverse	IN/ 000 –> 255	A003 [oper. 16 bits] ↓ Input number	A0030020 = IN/ 032
TIME DELAY	TIME 001 –> 999 TIME W_00 –> 15	A004[oper.4bits]0[oper.11bits] ↓ SAP Marker number ↓ Value in 1/10s A004 0000 1 [oper.11bits] ↓ Word number	A004000A = TIME 010 A004300A = TIME 010 Marker V03 A004080A = TIME W10 A004080F = TIME W15
BIT	BIT 000 –> 127 / BIT 000 –> 127	A005 [oper. 16 bits] A006 [oper. 16 bits] ↓ Bit number	A0050063 = BIT 99 A006007D = / BIT 127

* The actions and outputs replaced by text (e.g.: `part grip 1`) keep the same CODOP

Display	Codop (hexadecimal)	Examples
FUNCTIONS (FUNC)		
SPEED in % of the speed set in the parameters		
VEL . X 001 –> 100 VEL . Y 001 –> 100 VEL . Z 001 –> 100 VEL . B 001 –> 100 VEL . C 001 –> 100	B000[oper.4bits][oper.12bits] B001[oper.4bits][oper.12bits] B002[oper.4bits][oper.12bits] B003[oper.4bits][oper.12bits] B004[oper.4bits][oper.12bits] ↓ SAP Marker N° ↓ Value in 1/10s	B0000062 = VEL.X 098 B001000A = VEL.Y 010 B0020012 = VEL.Z 018 B0030064 = VEL.B 100 B004A032 = VEL.C 050 Marker V10
SPEED in mm/s programmed directly (or in °/s for a rotating axis)		
VEL . X ABS speed VEL . Y ABS speed VEL . Z ABS speed VEL . B ABS speed VEL . C ABS speed	B070[oper.4bits][oper.24bits] B071[oper.4bits][oper.24bits] B072[oper.4bits][oper.24bits] B073[oper.4bits][oper.24bits] B074[oper.4bits][oper.24bits] ↓ SAP Marker N° ↓ Value in 1/10s	B07003E8 = VEL.X ABS 1000.0 B07105DC = VEL.Y ABS 1500.0 B07207D0 = VEL.Z ABS 2000.0 B073005A = VEL.B ABS 90.0 B074002D = VEL.C ABS 45.0
SPEED in mm/s programmed in a WWORD (or in °/s for a rotating axis)		
VEL . X WW_*nn VEL . Y WW_*nn VEL . Z WW_*nn VEL . B WW_*nn VEL . C WW_*nn *(nn = 00 à 55 and 66 à 67)	B050 0000 [oper.12bits] B051 0000 [oper.12bits] B052 0000 [oper.12bits] B053 0000 [oper.12bits] B054 0000 [oper.12bits] ↓ wword N°	B0500042 = VEL.X WW066 B0510043 = VEL.Y WW067 B0520042 = VEL.Z WW066 B0530042 = VEL.B WW066 B0540043 = VEL.C WW067
SPEED linked to the mould speed		
VEL . X MOULD VEL . Y MOULD VEL . Z MOULD VEL . B MOULD VEL . C MOULD	B0C0 B0C1 B0C2 B0C3 B0C4	Reserved
SPEED linked to the ejector speed		
VEL . X EJECT VEL . Y EJECT VEL . Z EJECT VEL . B EJECT VEL . C EJECT	B0D0 B0D1 B0D2 B0D3 B0D4	Reserved

Display	Codop (hexadecimal)	Examples
ACCELERATION in % of the acceleration set in the parameters		
ACC . X 001 –> 100 ACC . Y 001 –> 100 ACC . Z 001 –> 100 ACC . B 001 –> 100 ACC . C 001 –> 100	B010 [oper. 16 bits] B011 [oper. 16 bits] B012 [oper. 16 bits] B013 [oper. 16 bits] B014 [oper. 16 bits] ↓ Value in %	B010000F = ACC.X 015 B0110064 = ACC.Y 100 B0120044 = ACC.Z 068 B0130005 = ACC.B 005 B0140032 = ACC.C 050
Master MOUVEMENT		
MASTER . X MASTER . Y MASTER . Z MASTER . B MASTER . C	B030 B031 B032 B033 B034	
IMPRECISION*		
IMP . X IMP . Y IMP . Z IMP . B IMP . C	B040[oper.4bits][oper.12bits] B041[oper.4bits][oper.12bits] B042[oper.4bits][oper.12bits] B043[oper.4bits][oper.12bits] B044[oper.4bits][oper.12bits] ↓ SAP Marker N° ↓ Value in 1/10 mm	B04031F4 = IMP X 50.0 I3 B04125DC = IMP Y 150.0 I2 B0420384 = IMP Z 90.0 B0430190 = IMP B 400.0 B0440DAC = IMP C 350.0
* The imprecise values must not be greater than 400.0 mm if they use an SAP marker.		
SLOW APPROACH in % of the maximum Slow Approach speed set in the parameters		
APL . X 001 –> 100 APL . Y 001 –> 100 APL . Z 001 –> 100 APL . B 001 –> 100 APL . C 001 –> 100	B020 [oper. 16 bits] B021 [oper. 16 bits] B022 [oper. 16 bits] B023 [oper. 16 bits] B024 [oper. 16 bits] ↓ Value in %	B0200026 = APL.X 026 B0210034 = APL.Y 034 B0220090 = APL.Z 090 B0230100 = APL.B 100 B0240010 = APL.C 010
Free MOVEMENT		
X . FREE Y . FREE Z . FREE B . FREE C . FREE	C040 C041 C042 C043 C044	
LINEARIZATION		
LIN.	B046	

Display	Codop (hexadecimal)	Examples
<u>MOTORIZED MOTIONS (Numerical operands)</u>		
ABSOLUTE		
X . ABS_L distance Y . ABS_L distance Z . ABS_L distance B . ABS_L distance C . ABS_L distance	C000[oper.8bits][oper.24bits] C001[oper.8bits][oper.24bits] C002[oper.8bits][oper.24bits] C003[oper.8bits][oper.24bits] C004[oper.8bits][oper.24bits]	C00000000664=X.ABS.L163.6 C001000F423F=Y.ABS.L99999.9 C00200000320=Z.ABS.L80.0 C0030000003F=B.ABS.L6.3 C0040000050C=C.ABS.L150.0
X . ABS_R angle Y . ABS_R angle Z . ABS_R angle B . ABS_R angle C . ABS_R angle	C100[oper.8bits][oper.24bits] C101[oper.8bits][oper.24bits] C102[oper.8bits][oper.24bits] C103[oper.8bits][oper.24bits] C104[oper.8bits][oper.24bits]	C10000000664=X.ABS.R00163.6 C101000005DC=Y.ABS.R00150.0 C10200000320=Z.ABS.R00080.0 C1030000003F=B.ABS.R00006.3 C10400000159=C.ABS.R00034.5
STACKING		
X . STK_L distance Y . STK_L distance Z . STK_L distance B . STK C . STK	C010[oper.8bits][oper.24bits] C011[oper.8bits][oper.24bits] C012[oper.8bits][oper.24bits] C053 C054	C01000008ACF=X.STK.L3453.5 C01100030DE3=Y.STK.L20016.3 C01200000159=Z.STK.L34.5 Reserved for general stackings Absolute distances from the header
X . STK_R angle Y . STK_R angle Z . STK_R angle	C110[oper.8bits][oper.24bits] C111[oper.8bits][oper.24bits] C112[oper.8bits][oper.24bits]	C11000008ACF=X.STK.R03453.5 C11100030DE3=Y.STK.R20016.3 C11200000159=Z.STK.R00034.5
RELATIVE		
X . REL_L distance Y . REL_L distance Z . REL_L distance B . REL_L distance C . REL_L distance	C020[oper.8bits][oper.24bits] C021[oper.8bits][oper.24bits] C022[oper.8bits][oper.24bits] C023[oper.8bits][oper.24bits] C024[oper.8bits][oper.24bits]	C020800000A0=X.REL.L–0016.0 C021000000A0=Y.REL.L–0016.0 C0228001869F=Z.REL.L–9999.9 C02300002706=B.REL.L+0999.9 C0240000000A=C.REL.L+0001.0
X . REL_R angle Y . REL_R angle Z . REL_R angle B . REL_R angle C . REL_R angle	C120[oper.8bits][oper.24bits] C121[oper.8bits][oper.24bits] C122[oper.8bits][oper.24bits] C123[oper.8bits][oper.24bits] C124[oper.8bits][oper.24bits] ↓ SAP Marker N° ↓ Value in 1/10 mm or 1/10°	C120000001C2=X.REL.R+45.0 C121800001C2=Y.REL.R–45.0 C122000000C8=Z.REL.R+20.0 C12380000159=B.REL.R–34.5 C1240000003F=C.REL.R+06.3

Display	Codop (hexadecimal)	Examples
CHECKING		
X . CTL_L distance Y . CTL_L distance Z . CTL_L distance B . CTL_L distance C . CTL_L distance	C030[oper.8bits][oper.24bits] C031[oper.8bits][oper.24bits] C032[oper.8bits][oper.24bits] C033[oper.8bits][oper.24bits] C034[oper.8bits][oper.24bits]	C03000000664=X.CTL.L00163.6 C031000F423F=Y.CTL.L9999.9 C03200000320=Z.CTL.L00080.0 C0330000003F=B.CTL.L00006.3 C0340500050C=C.CTL.L00150.0 Marker P05
X . CTL_R angle Y . CTL_R angle Z . CTL_R angle B . CTL_R angle C . CTL_R angle	C130[oper.8bits][oper.24bits] C131[oper.8bits][oper.24bits] C132[oper.8bits][oper.24bits] C133[oper.8bits][oper.24bits] C134[oper.8bits][oper.24bits] ↓ SAP Marker N° ↓ Value in 1/10 mm or 1/10°	C13000000664=X.CTL.R00163.6 C131000F423F=Y.CTL.R9999.9 C13200000320=Z.CTL.R00080.0 C1330000003F=B.CTL.R00006.3 C1340000050C=C.CTL.R00150.0
TEACHING		
⊔ |___| ⊔ Teach ↓ Previous instruction	|C___| [oper.8bits]AAAAAA ↓ Instruction code ↓ SAP Marker N°	C01000AAAAAA=X.STK.L Teach C10200AAAAAA=Z.ABS.R Teach
<u>MOTORIZED MOTIONS (Words)</u>		
ABSOLUTE		
X . ABS_L WW *nn Y . ABS_L WW *nn Z . ABS_L WW *nn B . ABS_L WW *nn C . ABS_L WW *nn	C200[oper.16bits] C201[oper.16bits] C202[oper.16bits] C203[oper.16bits] C204[oper.16bits]	C200000A = X.ABS.L WW10
X . ABS_R WW *nn Y . ABS_R WW *nn Z . ABS_R WW *nn B . ABS_R WW *nn C . ABS_R WW *nn	C300[oper.16bits] C301[oper.16bits] C302[oper.16bits] C303[oper.16bits] C304[oper.16bits]	C300000A = X.ABS.R WW10
STACKING		
X . STK_L WW *nn Y . STK_L WW *nn Z . STK_L WW *nn	C210[oper.16bits] C211[oper.16bits] C212[oper.16bits]	C210000B = X.STK.L WW11
X . STK_R WW *nn Y . STK_R WW *nn Z . STK_R WW *nn	C310[oper.16bits] C311[oper.16bits] C312[oper.16bits]	C3100020 = X.STK.R WW32

Display	Codop (hexadecimal)	Examples
RELATIVE		
X . REL_L WW *nn Y . REL_L WW *nn Z . REL_L WW *nn B . REL_L WW *nn C . REL_L WW *nn	C220[oper.16bits] C221[oper.16bits] C222[oper.16bits] C223[oper.16bits] C224[oper.16bits]	C2200041 = X.REL.L WW65
X . REL_R WW *nn Y . REL_R WW *nn Z . REL_R WW *nn B . REL_R WW *nn C . REL_R WW *nn	C320[oper.16bits] C321[oper.16bits] C322[oper.16bits] C323[oper.16bits] C324[oper.16bits]	C3200001 = X.REL.R WW01
CHECKING		
X . CTL_L WW *nn Y . CTL_L WW *nn Z . CTL_L WW *nn B . CTL_L WW *nn C . CTL_L WW *nn	C230[oper.16bits] C231[oper.16bits] C232[oper.16bits] C233[oper.16bits] C234[oper.16bits]	C2300010 = X.CTL.L WW16
X . CTL_R WW *nn Y . CTL_R WW *nn Z . CTL_R WW *nn B . CTL_R WW *nn C . CTL_R WW *nn	C330[oper.16bits] C331[oper.16bits] C332[oper.16bits] C333[oper.16bits] C334[oper.16bits]	C3300041 = X.CTL.R WW65

↓ WWord N°

* nn = 00 –> 55 and 64 –> 65

Display	Codop (hexadecimal)	Examples
MOTORIZED MOTIONS (cont'd)		
POS ANA		
X = POS ANA + distance Y = POS ANA + distance Z = POS ANA + distance B = POS ANA + distance C = POS ANA + distance	C060[oper.32bits] C061[oper.32bits] C062[oper.32bits] C063[oper.32bits] C064[oper.32bits]	
X = POS ANA + angle Y = POS ANA + angle Z = POS ANA + angle B = POS ANA + angle C = POS ANA + angle	C160[oper.32bits] C161[oper.32bits] C162[oper.32bits] C163[oper.32bits] C164[oper.32bits]	
POS NUM		
X = POS NUM + distance Y = POS NUM + distance Z = POS NUM + distance B = POS NUM + distance C = POS NUM + distance	C070[oper.32bits] C071[oper.32bits] C072[oper.32bits] C073[oper.32bits] C074[oper.32bits]	
X = POS NUM + angle Y = POS NUM + angle Z = POS NUM + angle B = POS NUM + angle C = POS NUM + angle	C170[oper.32bits] C171[oper.32bits] C172[oper.32bits] C173[oper.32bits] C174[oper.32bits]	
VEL ANA INTEGRAL		
X = VEL ANA Y = VEL ANA Z = VEL ANA B = VEL ANA C = VEL ANA	C090 C091 C092 C093 C094	Linear axis
X = VEL ANA Y = VEL ANA Z = VEL ANA B = VEL ANA C = VEL ANA	C190 C191 C192 C193 C194	Rotating axis
VEL NUM NORMAL		
X = VEL NUM_N Y = VEL NUM_N Z = VEL NUM_N B = VEL NUM_N C = VEL NUM_N	C0A0 C0A1 C0A2 C0A3 C0A4	
X = VEL NUM_N Y = VEL NUM_N Z = VEL NUM_N B = VEL NUM_N C = VEL NUM_N	C1A0 C1A1 C1A2 C1A3 C1A4	

Display	Codop (hexadecimal)	Examples
VEL NUM NORMAL		
X = VEL NUM_N Y = VEL NUM_N Z = VEL NUM_N B = VEL NUM_N C = VEL NUM_N	C0A0 C0A1 C0A2 C0A3 C0A4	
X = VEL NUM_N Y = VEL NUM_N Z = VEL NUM_N B = VEL NUM_N C = VEL NUM_N	C1A0 C1A1 C1A2 C1A3 C1A4	
VEL NUM INTEGRAL		
X = VEL NUM_I Y = VEL NUM_I Z = VEL NUM_I B = VEL NUM_I C = VEL NUM_I	C0B0 C0B1 C0B2 C0B3 C0B4	
X = VEL NUM_I Y = VEL NUM_I Z = VEL NUM_I B = VEL NUM_I C = VEL NUM_I	C1B0 C1B1 C1B2 C1B3 C1B4	
TEACHING		
⊔ |___| ⊔ Teach ↓ Previous Instruction	|C___| [oper.8bits]AAAAAA ↓ ↓ Instruction code / SAP Marker N°	C16000AAAAAA=X.POS ANA + Teach C17200AAAAAA=Z.POS NUM + Teach
Teaching is possible for the POS ANA and POS NUM instructions.		
POS MOULD		
X = POS MOULD + distance Y = POS MOULD + distance Z = POS MOULD + distance B = POS MOULD + distance C = POS MOULD + distance	C0C0[oper.32bits] C0C1[oper.32bits] C0C2[oper.32bits] C0C3[oper.32bits] C0C4[oper.32bits]	
POS EJECT		
X = POS EJECT + distance Y = POS EJECT + distance Z = POS EJECT + distance B = POS EJECT + distance C = POS EJECT + distance	C0D0[oper.32bits] C0D1[oper.32bits] C0D2[oper.32bits] C0D3[oper.32bits] C0D4[oper.32bits]	

Display	Codop (hexadecimal)	Examples
<u>IF INSTRUCTION</u> ONE OPERAND		
IF BIT 000 –> 127	D000 [oper. 16 bits]	
IF /BIT 000 –> 127	D010 [oper. 16 bits]	
IF OUT 000 –> 255	D001 [oper. 16 bits]	
IF/OUT 000 –> 255	D011 [oper. 16 bits]	
IF IN/000 –> 255	D002 [oper. 16 bits]	
IF IN 000 –> 255	D003 [oper. 16 bits]	
IF/IN 000 –> 255	D013 [oper. 16 bits]	
IF TIM 00 –> 15	D004 [oper. 16 bits]	
IF/TIM 00 –> 15	D014 [oper. 16 bits]	
	↓ Operand number	
<u>IF INSTRUCTION</u> TWO OPERANDS : WORD		
IF WRD 000 –> 4095	D300 [oper. 16 bits]	
IF /WRD 000 –> 4095	D310 [oper. 16 bits]	
= 0 –> 9999	D400 [oper. 16 bits]	
> = 0 –> 9999	D401 [oper. 16 bits]	
< = 0 –> 9999	D402 [oper. 16 bits]	
AND 0 –> 9999	D403 [oper. 16 bits]	**<u>Note</u>** : If the decimal value cannot exceed 9,999, the hexadecimal value goes up to 65,535.
= 0 –> FFFF	D410 [oper. 16 bits]	
> = 0 –> FFFF	D411 [oper. 16 bits]	
< = 0 –> FFFF	D412 [oper. 16 bits]	
AND 0 –> FFFF	D413 [oper. 16 bits]	
= CNT 00 –> 15	D420 [oper. 16 bits]	
> = CNT 00 –> 15	D421 [oper. 16 bits]	
< = CNT 00 –> 15	D422 [oper. 16 bits]	
AND CNT 00 –> 15	D423 [oper. 16 bits]	
= IN 000 –> 240	D430 [oper. 16 bits]	
> = IN 000 –> 240	D431 [oper. 16 bits]	
< = IN 000 –> 240	D432 [oper. 16 bits]	
AND IN 000 –> 240	D433 [oper. 16 bits]	
= WRD 0000 –> 4095	D440 [oper. 16 bits]	
> = WRD 0000 –> 4095	D441 [oper. 16 bits]	
< = WRD 0000 –> 4095	D442 [oper. 16 bits]	
AND WRD 0000 –> 4095	D443 [oper. 16 bits]	

Display	Codop (hexadecimal)	Examples
IF INSTRUCTION TWO OPERANDS : CNT		
IF CNT 000 –> 15	D340 [oper. 16 bits]	
IF /CNT 000 –> 15	D350 [oper. 16 bits]	
= 0 –> 9999	D900 [oper. 32 bits]	
> = 0 –> 9999	D901 [oper. 32 bits]	
< = 0 –> 9999	D902 [oper. 32 bits]	
AND 0 –> 9999	D903 [oper. 32 bits]	
= 0 –> FFFF	D910 [oper. 32 bits]	
> = 0 –> FFFF	D911 [oper. 32 bits]	
< = 0 –> FFFF	D912 [oper. 32 bits]	
AND 0 –> FFFF	D913 [oper. 32 bits]	
= CNT 00 –> 15	D920 [oper. 16 bits]	
> = CNT 00 –> 15	D921 [oper. 16 bits]	
< = CNT 00 –> 15	D922 [oper. 16 bits]	
AND CNT 00 –> 15	D923 [oper. 16 bits]	
= IN 000 –> 240	D930 [oper. 16 bits]	
> = IN 000 –> 240	D931 [oper. 16 bits]	
< = IN 000 –> 240	D932 [oper. 16 bits]	
AND IN 000 –> 240	D933 [oper. 16 bits]	
= WRD 0000 –> 4095	D940 [oper. 16 bits]	
> = WRD 0000 –> 4095	D941 [oper. 16 bits]	
< = WRD 0000 –> 4095	D942 [oper. 16 bits]	
AND WRD 0000 –> 4095	D943 [oper. 16 bits]	

Display	Codop (hexadecimal)	Examples
<u>IF INSTRUCTION</u> TWO OPERANDS : WWORD		
IF WWRD 000 –> 127 IF /WWRD 000 –> 127	D320 [oper. 16 bits] D330 [oper. 16 bits]	
= 0 –> 9999999 > = 0 –> 9999999 < = 0 –> 9999999 AND 0 –> 9999999	D500 [oper. 32 bits] D501 [oper. 32 bits] D502 [oper. 32 bits] D503 [oper. 32 bits]	**<u>Note</u>** : If the decimal value cannot exceed 9,999,999, the hexadecimal value goes up to 4,294,967,295.
= 0 –> FFFFFFFF > = 0 –> FFFFFFFF < = 0 –> FFFFFFFF AND 0 –> FFFFFFFF	D510 [oper. 32 bits] D511 [oper. 32 bits] D512 [oper. 32 bits] D513 [oper. 32 bits]	
= CNT 00 –> 15 > = CNT 00 –> 15 < = CNT 00 –> 15 AND CNT 00 –> 15	D520 [oper. 16 bits] D521 [oper. 16 bits] D522 [oper. 16 bits] D523 [oper. 16 bits]	
= IN 000 –> 240 > = IN 000 –> 240 < = IN 000 –> 240 AND IN 000 –> 240	D530 [oper. 16 bits] D531 [oper. 16 bits] D532 [oper. 16 bits] D533 [oper. 16 bits]	
= WRD 0000 –> 4095 > = WRD 0000 –> 4095 < = WRD 0000 –> 4095 AND WRD 0000 –> 4095	D540 [oper. 16 bits] D541 [oper. 16 bits] D542 [oper. 16 bits] D543 [oper. 16 bits]	
= WWRD 0000 –> 127 > = WWRD 0000 –> 127 < = WWRD 0000 –> 127 AND WWRD 0000 –> 127	D550 [oper. 16 bits] D551 [oper. 16 bits] D552 [oper. 16 bits] D553 [oper. 16 bits]	
<u>INCREMENTATION / DECREMENTATION</u>		
INC CNT 00 –> 15 INC CNT 0041 –> 9980	D01B 00[oper. 8 bits] D01B [oper. 8 bits][oper. 8 bits] ↓ PRG number ↓ SP number	
DEC CNT 00 –> 15 DEC 0041 –> 9980	D01C 00[oper. 8 bits] D01C [oper. 8 bits][oper. 8 bits] ↓ PRG number ↓ SP number	

Display	Codop (hexadecimal)	Examples
<u>INITIALIZATION</u> ONE OPERAND		
SET BIT 032 –> 127	D015 [oper. 16 bits]	
RST BIT 032 –> 127	D017 [oper. 16 bits]	
SET OUT 000 –> 255	D016 [oper. 16 bits]	
RST OUT 000 –> 255	D018 [oper. 16 bits]	
RST WRD 0000 –> 4095	D019 [oper. 16 bits]	
RST WWRD 0000 –> 63	D01D [oper. 16 bits]	
RST CNT 00 –> 15	D01A 00[oper. 8 bits]	
RST CNT 0041 –> 9980	D01A [oper. 8 bits][oper. 8 bits] ↓ PRG number ↓ SP number	
<u>INITIALIZATION</u> TWO OPERANDS : WORD		
SET WRD 000 –> 4095	D600 [oper. 16 bits]	
= 0 –> 9999	D700 [oper. 16 bits]	
+ 0 –> 9999	D701 [oper. 16 bits]	
– 0 –> 9999	D702 [oper. 16 bits]	
x 0 –> 9999	D703 [oper. 16 bits]	
/ 0 –> 9999	D704 [oper. 16 bits]	
AND 0 –> 9999	D705 [oper. 16 bits]	
OR 0 –> 9999	D706 [oper. 16 bits]	
= 0 –> FFFF	D710 [oper. 16 bits]	
+ 0 –> FFFF	D711 [oper. 16 bits]	
– 0 –> FFFF	D712 [oper. 16 bits]	
x 0 –> FFFF	D713 [oper. 16 bits]	
/ 0 –> FFFF	D714 [oper. 16 bits]	
AND 0 –> FFFF	D715 [oper. 16 bits]	
OR 0 –> FFFF	D716 [oper. 16 bits]	
= CNT 00 –> 15	D720 [oper. 16 bits]	
+ CNT 00 –> 15	D721 [oper. 16 bits]	
– CNT 00 –> 15	D722 [oper. 16 bits]	
x CNT 00 –> 15	D723 [oper. 16 bits]	
/ CNT 00 –> 15	D724 [oper. 16 bits]	
AND CNT 00 –> 15	D725 [oper. 16 bits]	
OR CNT 00 –> 15	D726 [oper. 16 bits]	
= IN 000 –> 240	D730 [oper. 16 bits]	
+ IN 000 –> 240	D731 [oper. 16 bits]	
– IN 000 –> 240	D732 [oper. 16 bits]	
x IN 000 –> 240	D733 [oper. 16 bits]	
/ IN 000 –> 240	D734 [oper. 16 bits]	
AND IN 000 –> 240	D735 [oper. 16 bits]	
OR IN 000 –> 240	D736 [oper. 16 bits]	
= WRD 0000 –> 4095	D740 [oper. 16 bits]	
+ WRD 0000 –> 4095	D741 [oper. 16 bits]	
– WRD 0000 –> 4095	D742 [oper. 16 bits]	
x WRD 0000 –> 4095	D743 [oper. 16 bits]	
/ WRD 0000 –> 4095	D744 [oper. 16 bits]	
AND WRD 0000 –> 4095	D745 [oper. 16 bits]	
OR WRD 0000 –> 4095	D746 [oper. 16 bits]	

Display	Codop (hexadecimal)	Examples
INITIALIZATION TWO OPERANDS : WWORD		
SET WWRD 000 –> 127	D620 [oper. 16 bits]	
= 0 –> 9999999	D800 [oper. 32 bits]	
+ 0 –> 9999999	D801 [oper. 32 bits]	
– 0 –> 9999999	D802 [oper. 32 bits]	
x 0 –> 9999999	D803 [oper. 32 bits]	
/ 0 –> 9999999	D804 [oper. 32 bits]	
AND 0 –> 9999999	D805 [oper. 32 bits]	
OR 0 –> 9999999	D806 [oper. 32 bits]	
= 0 –> FFFFFFFF	D810 [oper. 32 bits]	
+ 0 –> FFFFFFFF	D811 [oper. 32 bits]	
– 0 –> FFFFFFFF	D812 [oper. 32 bits]	
x 0 –> FFFFFFFF	D813 [oper. 32 bits]	
/ 0 –> FFFFFFFF	D814 [oper. 32 bits]	
AND 0 –> FFFFFFFF	D815 [oper. 32 bits]	
OR 0 –> FFFFFFFF	D816 [oper. 32 bits]	
= CNT 00 –> 15	D820 [oper. 16 bits]	
+ CNT 00 –> 15	D821 [oper. 16 bits]	
– CNT 00 –> 15	D822 [oper. 16 bits]	
x CNT 00 –> 15	D823 [oper. 16 bits]	
/ CNT 00 –> 15	D824 [oper. 16 bits]	
AND CNT 00 –> 15	D825 [oper. 16 bits]	
OR CNT 00 –> 15	D826 [oper. 16 bits]	
= IN 000 –> 240	D830 [oper. 16 bits]	
+ IN 000 –> 240	D831 [oper. 16 bits]	
– IN 000 –> 240	D832 [oper. 16 bits]	
x IN 000 –> 240	D833 [oper. 16 bits]	
/ IN 000 –> 240	D834 [oper. 16 bits]	
AND IN 000 –> 240	D835 [oper. 16 bits]	
OR IN 000 –> 240	D836 [oper. 16 bits]	
= WRD 0000 –> 4095	D840 [oper. 16 bits]	
+ WRD 0000 –> 4095	D841 [oper. 16 bits]	
– WRD 0000 –> 4095	D842 [oper. 16 bits]	
x WRD 0000 –> 4095	D843 [oper. 16 bits]	
/ WRD 0000 –> 4095	D844 [oper. 16 bits]	
AND WRD 0000 –> 4095	D845 [oper. 16 bits]	
OR WRD 0000 –> 4095	D846 [oper. 16 bits]	
= WWRD 0000 –> 127*	D850 [oper. 16 bits]	* also possible with WWORD 200 –> 202
+ WWRD 0000 –> 127	D851 [oper. 16 bits]	
– WWRD 0000 –> 127	D852 [oper. 16 bits]	
x WWRD 0000 –> 127	D853 [oper. 16 bits]	
/ WWRD 0000 –> 127	D854 [oper. 16 bits]	
AND WWRD 0000 –> 127	D855 [oper. 16 bits]	
OR WWRD 0000 –> 127	D856 [oper. 16 bits]	

Display	Codop (hexadecimal)	Examples
INITIALIZATION TWO OPERANDS : CNT		
SET CNT 00 –> 15	D640 00[oper. 8 bits]	Standard counter
SET CNT 0041 –> 9980	D640 [oper. 8 bits][oper. 8 bits] (first: PRG number; second: SP number)	Stacking counter
= 0 –> 9999	DA00 [oper. 16 bits]	
+ 0 –> 9999	DA01 [oper. 16 bits]	
– 0 –> 9999	DA02 [oper. 16 bits]	
x 0 –> 9999	DA03 [oper. 16 bits]	
/ 0 –> 9999	DA04 [oper. 16 bits]	
AND 0 –> 9999	DA05 [oper. 16 bits]	
OR 0 –> 9999	DA06 [oper. 16 bits]	
= 0 –> FFFF	DA10 [oper. 16 bits]	
+ 0 –> FFFF	DA11 [oper. 16 bits]	
– 0 –> FFFF	DA12 [oper. 16 bits]	
x 0 –> FFFF	DA13 [oper. 16 bits]	
/ 0 –> FFFF	DA14 [oper. 16 bits]	
AND 0 –> FFFF	DA15 [oper. 16 bits]	
OR 0 –> FFFF	DA16 [oper. 16 bits]	
= CNT 00 –> 15	D920 [oper. 16 bits]	
+ CNT 00 –> 15	D921 [oper. 16 bits]	
– CNT 00 –> 15	D922 [oper. 16 bits]	
x CNT 00 –> 15	D923 [oper. 16 bits]	
/ CNT 00 –> 15	D924 [oper. 16 bits]	
AND CNT 00 –> 15	D925 [oper. 16 bits]	
OR CNT 00 –> 15	D926 [oper. 16 bits]	
= IN 000 –> 240	DA30 [oper. 16 bits]	
+ IN 000 –> 240	DA31 [oper. 16 bits]	
– IN 000 –> 240	DA32 [oper. 16 bits]	
x IN 000 –> 240	DA33 [oper. 16 bits]	
/ IN 000 –> 240	DA34 [oper. 16 bits]	
AND IN 000 –> 240	DA35 [oper. 16 bits]	
OR IN 000 –> 240	DA36 [oper. 16 bits]	
= WRD 0000 –> 4095	DA40 [oper. 16 bits]	
+ WRD 0000 –> 4095	DA41 [oper. 16 bits]	
– WRD 0000 –> 4095	DA42 [oper. 16 bits]	
x WRD 0000 –> 4095	DA43 [oper. 16 bits]	
/ WRD 0000 –> 4095	DA44 [oper. 16 bits]	
AND WRD 0000 –> 4095	DA45 [oper. 16 bits]	
OR WRD 0000 –> 4095	DA46 [oper. 16 bits]	

II – 2. PLC programs

Type of instruction	Display	Codop (hexadecimal)
PROG.PLC xx header (num)	PLC xx	FC [oper. 16 bits] ↓ PLC number
TEST CONDITION	IF ...	See part programs
INITIALIZATION	SET ... RST ... INC ... DEC ...	See part programs
COMPARISON CNT xxxx > = xxxx	CMP CNT 00 –> 15 VAL 0000 –> FFFF CMP CNT 0041 –> 9980 VAL 0000 –> FFFF	D020 [oper. 16 bits][oper. 16 bits] ↓ Counter number ↓ Value
TIMER xx VALUE xxxx	TIMER 00 –> 15 VAL 0 –> 9999	D021 [oper. 16 bits][oper. 16 bits] ↓ Timer number ↓ preselection value
AND FUNCTION on BIT	AND BIT 000 –> 127	D022 [oper. 16 bits]
AND FUNCTION on OUT-PUT	AND OUT 000 –> 255	D023 [oper. 16 bits]
OR FUNCTION on BIT	OR BIT 000 –> 127	D024 [oper. 16 bits]
OR FUNCTION on OUT-PUT	OR OUT 000 –> 255	D025 [oper. 16 bits]
END OF PROGRAM	END	F5 [oper. 16 bits] ↓ PLC number

III – PROGRAM CODES

III – 1. Declaration of programs, subroutines and PLCs

▶ Header codes of PRG, SP,..., SR, PLC

- F9b xn = Main program
- b = 0, standard PRG (encoded on 15 bits)
  b = 1 , SAP PRG (encoded on 15 bits)
- FAnn = STD, STK.. // subroutine (see stacking header)
- FBnn = Return subroutine (see home return header)
- FCnn = PLC program
- FEnn = FREE

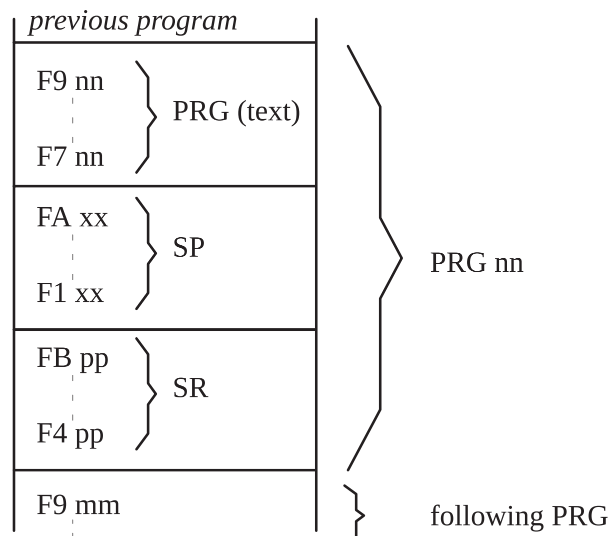
▶ STEP TRANSITION codes

- EC00 + Step number 0 to 999
- E.g. : EC12 => Step number 18 (decimal)
- E.g. : ED00 => Step number 256 (decimal)

▶ END of PRG, SP..., SR, PLC codes

- F0nn = End of "standard" SP nn.
- F1nn = End of "standard" stacking SP nn.
- F2nn = End of "general" stacking SP nn.
- F3nn = End of SP // nn.
- F4nn = End of simple or total SR nn.
- F8nn = End of simple or total SR with return to step 0 of PRG 00.
- F5nn = End of PLC nn.
- F7nn = End of main program (PRG) nn.

▶ PRG architecture in the memory area



III – 2. <u>Subroutine and program calls</u>

▶ <u>SPECIFIC codes for SP, SR, PLC as an instruction</u>

- E000 [oper. 16 bits] :

Standard SP SP nn Lmm (nn = 01 to 40) (mm = 00 to 99)

Regular Stacking SP SP nn D Lmm (or I Lmm) (nn = 41 to 60) (mm = 00 to 99)

General Stacking SP SP nn D Lmm (or I Lmm) (nn = 61 to 80) (mm = 00 to 99)

Parallel SP SP nn L00 (nn = 81 to 99)

The operand contains :

. high order word –> the LABEL number

–> bit 0 x 8000 at 0 indicates DIRECT

–> bit 0 x 8000 at 1 indicates REVERSE

. low order word –> the SP number.

E.g. : E000 0103 –> SP 03 L01

E.g. : E000 8229 –> SP 41 I L02

- E100 [oper. 16 bits] : PLC prog. – Display : PLC 00 (to 99)
- E500 [oper. 16 bits] : Home Return – Display : SR 01 (to 99)

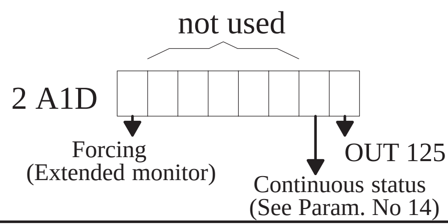
▶ <u>Return label</u>

- E600 [oper. 16 bits] : Labels ”L” for SP – Display : L00 to L99
- E700 [oper. 16 bits] : Labels ”R” for SR – Display : R00 to R99

IV – VARIABLES' ADDRESSES

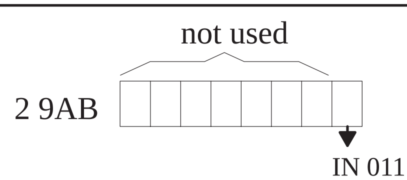
IV – 1. Output – OUT –

Accessible in read and write.

Number (logical address)	Physical address	Structures / Functions
OUT 000 ↓ OUT 255	28A0 ↓ 299F	

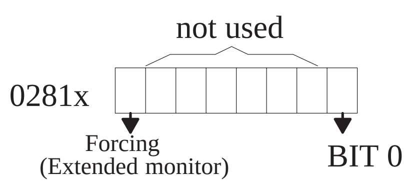
IV – 2. Input – IN –

Accessible in read.

Number (logical address)	Physical address	Structures / Functions
IN 000 ↓ IN 255	29A0 ↓ 2A9F	

IV – 3. User and system bits – BIT –

Each address corresponds to an 8 bit structure in memory.



x = bit number in hexadecimal (e.g.: Bit 31, address = 0282F).
Only the low order word is used.

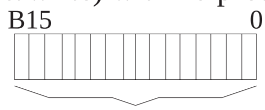
– System bits accessible in Read – No. 0 to 30.

– System bits accessible in Read and Write – No. 31 to 33.

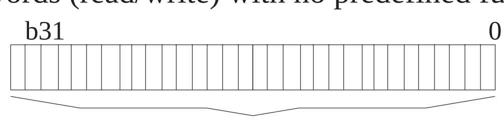
– User bits accessible in Read and Write – No. 34 to 127.

For the definition of these bits, see the Programming Level 2 manual, paragraph I3.

IV – 4. 16 bits user and system words – WRD –

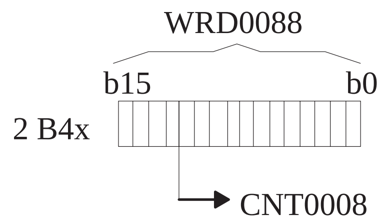
Number (logical address)	Physical address	Structures / Functions
WRD 0000 ↓ WRD 0031	2AA0 ↓ 2ADF	32 user Words (read/write) with no predefined functions.  16 bit structure available
WRD 0032 ↓ WRD 0063	2AE0 ↓ 2B1E	32 system Words (read only). For the definition of these words, see the Programming Level 2 manual, paragraph I4
WRD 0064 ↓ WRD 0079	2B20 ↓ 2B3F	16 user Words (read/write) supporting the PLC timers (TIM 00 to TIM 15).
WRD 0080 ↓ WRD 0095	2B40 ↓ 2B5F	16 user Words (read/write) supporting the standard counters (CNT 00 to CNT 15).
WRD 0096 WRD 4096	2B60 3A9F	4000 user Words (read/write) supporting the stacking subroutine counters (CNT 0041 to CNT 9980).

IV – 5. 32 bit user and system words – WWRD –

Number (logical address)	Physical address	Structures / Functions
WWRD 000 ↓ WWRD 063	6230 ↓ 6327	64 user Words (read/write) with no predefined functions.  32 bit structure available
WWRD 064 ↓ WWRD 127	6328 ↓ 642C	64 system Words (read only). For the definition of these words, see the Programming Level 2 manual, paragraph I5
WWRD 0116 WWRD 0117	6400 6404	*Specific words* Values for calculating the automatic anticipated restart. Values for calculating the automatic anticipated restart. See chapter VI – page 29.

IV – 6. Counters

Each address corresponds to a 16 bit structure in the memory.



. values from 0000 to 9999 in decimal

. values from 0000 to FFFF in hexadecimal

x = bit number in hexadecimal (e.g.: CNT 0008, address = 2 B50).

– Standard counters – No. 0000 to 0015 (0x2B40 to 0x2B5E).

– Regular stacking counters – No. 0041 to 9960 (as from 0x2 B60).

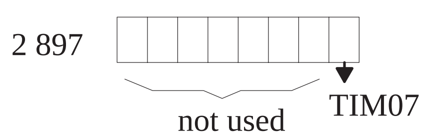
– General stacking counters – No 0061 to 9980.

For the definition of these counters, see the Programming Level 2 manual, paragraph I6.

IV – 7. Timers

IV – 7. 1.End of timer for part program

Accessible in read and write.

Number (logical address)	Physical address	Structures / Functions
TIM00	2 890	 Only the low order word is used
TIM01	2 891	
TIM02	2 892	
TIM03	2 893	
TIM04	2 894	
TIM05	2 895	
TIM06	2 896	
TIM07	2 897	
TIM08	2 898	
TIM09	2 899	
TIM10	2 89A	
TIM11	2 89B	
TIM12	2 89C	
TIM13	2 89D	
TIM14	2 89E	
TIM15	2 89F	

IV – 7. 2.End of timer for PLC

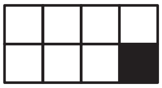
TIM00 to 15 = WRD 0064 to 0079 see chapter .

Accessible in read and write.

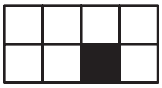
V – CPU FAULT SIGNALLING

V – 1. Flashing LEDs

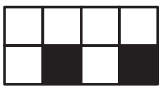
These signal a CAN network fault by displaying the problem number in binary on the LEDs at the bottom of the CPU, and the node number (if concerned) on the LEDs at the top if the pendant is not functioning.

3  0

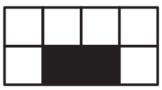
1 = CAN driver initialization fault



2 = Write problem in Flashprom

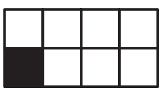


5 = A double (or more) node on the network (code + node)

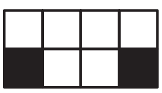


6 = Problem during the CONNECTION phase (code + node)

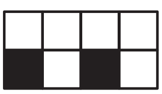
7 = Problem during the PREPARATION phase (code + node)



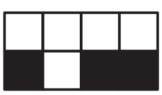
8 = Problem during the START phase (code + node)



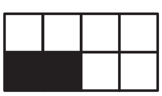
9 = The network does not correspond to the parametered configuration (code + node)



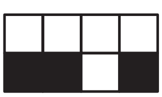
10 = "Node–guarding" problem (code + node). Communication fault with the pendant ; this may be due to the CAN speed being too great for the length of the cable used, or a bad line adaptation, or interference, etc.



11 = CPU emission problem



12 = CPU reception problem



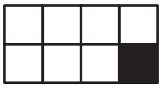
13 = Topology fault of the remote I/O

15 = EMERGENCY message received (code + node). Problem on the pendant or with communication between the pendant and the CPU (see 10)

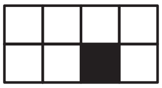
Note : In the event of a NODE GUARDING fault, fault 15 may appear alternately with fault 10.

V – 2. Fixed LEDs

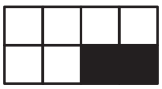
These signal a fault when powering up by giving the problem number in binary on the LEDs at the bottom of the CPU, and the node number (if concerned) on the LEDs at the top if the pendant is not functioning.



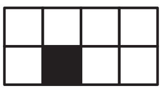
1 = Problem with recovering the parameters in Flashprom



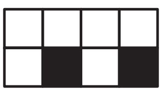
2 = Problem during the opening of the PC link



3 = Problem during the opening of the EUROMAP 17 link



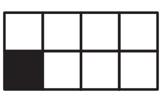
4 = Problem during the opening of the printer 2 link



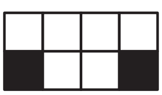
5 = Problem during the opening of the CAN link

6 = Message not present in Flashprom

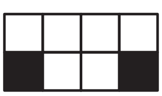
7 = Problem with the CPU's RAM



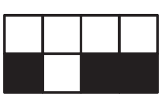
8 = Problem with the Flashprom's checksum



9 = Problem with the axes defined and the axes' boards present



10 = The configuration has changed



11 = Problem during the initialization of the axes' boards by the CPU

15 = Communication problem with the pendant during powering up. The CAN speed may be changed by transferring the parameters with the PC at 2400 Bds, slave = 1.

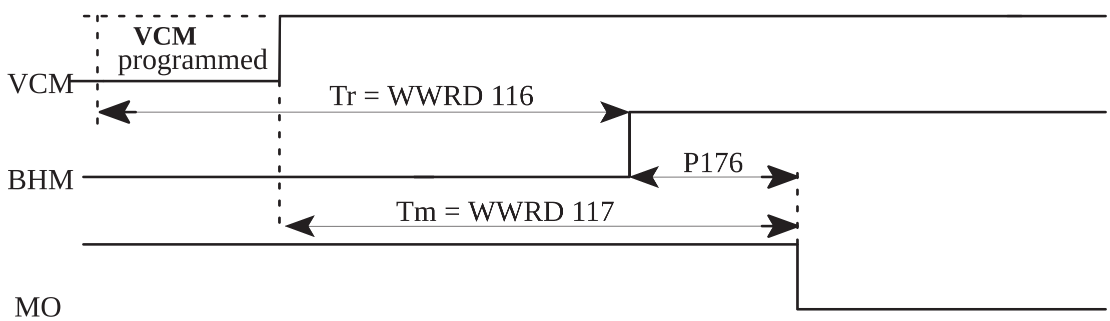
VI – IMM ANTICIPATED RESTART

- ▶ Parameter 174 : type of IMM anticipated restart
  - 0 : no anticipated restart
  - 1 : anticipated restart
  - 2 : programmed delay anticipated restart –> WWRD 63 programmed in step 0.
- ▶ Parameter 175 : basic value of the auto–adaptative delay and double the minimum value of the programmed delay
- ▶ Parameter 176 : minimum value of the auto–adaptative delay (safety margin)

Anticipated restart effective if :

- offset wait is not valid (parameter 451)
- and if the robot is in automatic
- and if Kv equals 100 %
- and if there is a SET WWRD63 in step 0 of the program
- and if the value of WWRD63 is greater than or equal to $\frac{\text{parameter 175}}{2}$

} in the case of restart with programmed delay (applies to the last two conditions)

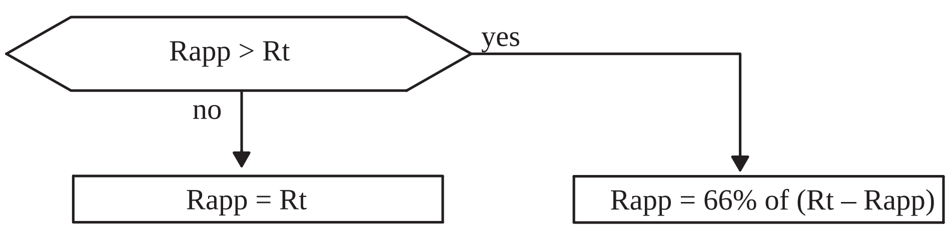


Tr = robot disengaging time in 1/10 s (WWRD 116)

Tm = IMM motion start time in 1/10 s (WWRD 117)

Rt = theoretical delay = Tr – Tm + P176 or 0 if the result is negative

Rapp = Applied delay



There is a fault if mould open (or OPA) goes to 0 and BHM = 0

`D_5 : MOVEMENT OUTSIDE CAMS` (if there is no anticipated restart running)

`D_32: PREMATURE MACHINE RESTART` (if there is an anticipated restart running)

▶ Safety circuit principle.

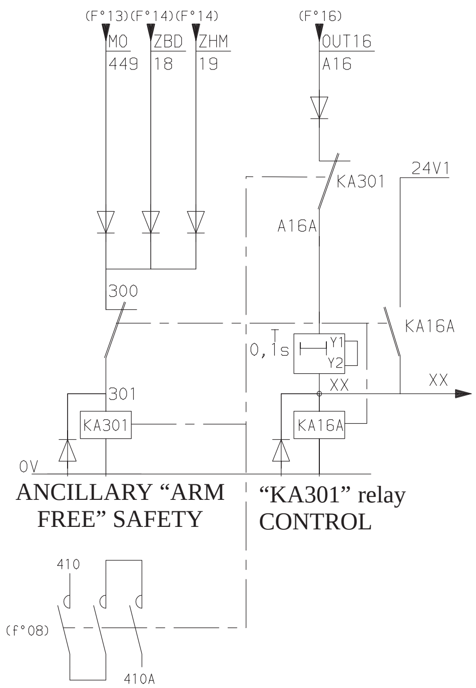
A hard–wired circuit controls the respective positions of the moving mould (“MO” = Mould Open signal) and of the robot (“ZBD” = Arm Free Area / “ZHM” = Outside Mould Area signal).

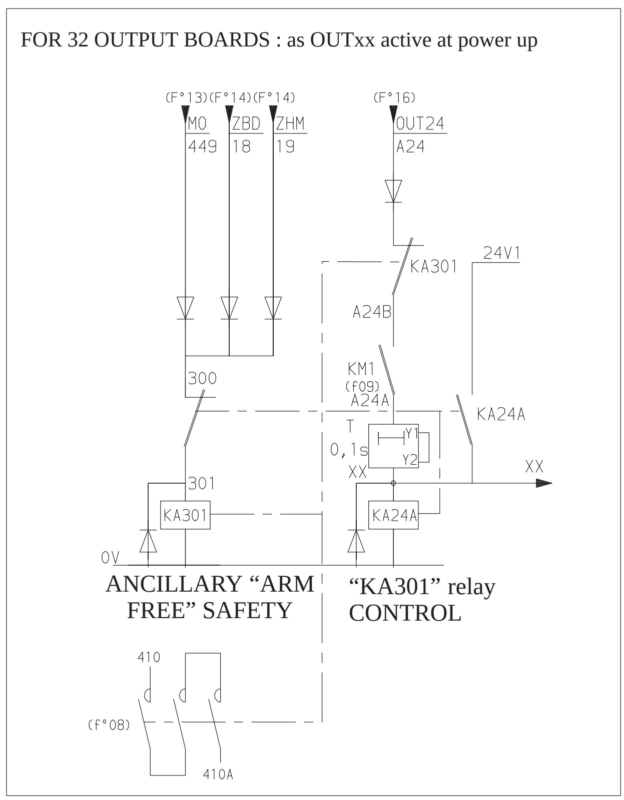
The output of this hard–wired circuit (”MO” + ”ZBD” + ”ZHM” = ”KA301”) activates a power relay (KA301 contactor).

During normal operation, the KA301 relay is activated. The KA301 contacts are used in series with the SBD relay contact from the interface board, which therefore means that the software safety that manages the SBD relay with a hard–wired safety device is doubled.

When there is a fault (robot position not conform compared to the moving mould position), the KA301 relay falls, which in turn activates the control relay KA16A, which is self–powered and stops the KA301 relay becoming active (the blocking of KA301 prohibits the IMM cycle).

You must power the robot cabinet down to cancel this fault.





V 2.0 Robot Enter the XX input number that controls the KA 301 relay in parameter 499.
If the input defined in this parameter goes to 1, the following fault message is displayed.

```
D_35: ANTICIPATED RESTART NOT CONFORM
```

We're Here to Help

Conair has made the largest investment in customer support in the plastics industry. Our service experts are available to help with any problem you might have installing and operating your equipment. Your Conair sales representative also can help analyze the nature of your problem, assuring that it did not result from misapplication or improper use.

How to Contact Customer Service

To contact Customer Service personnel, call:

PARTS & SERVICE 800 458 1960
Instant Access CONAIR™

From outside the United States, call: 814-437-6861

You can commission Conair service personnel to provide on-site service by contacting the Customer Service Department. Standard rates include an on-site hourly rate, with a one-day minimum plus expenses.

Before You Call ...

If you do have a problem, please complete the following checklist before calling Conair:

- ❒ Make sure you have all model, serial and parts list numbers for your particular equipment. Service personnel will need this information to assist you.
- ❒ Make sure power is supplied to the equipment.
- ❒ Make sure that all connectors and wires within and between loading control and related components have been installed correctly.
- ❒ Check the troubleshooting guide of this manual for a solution.
- ❒ Thoroughly examine the instruction manual(s) for associated equipment, especially controls. Each manual may have its own troubleshooting guide to help you.
- ❒ Check that the equipment has been operated as described in this manual.
- ❒ Check accompanying schematic drawings for information on special considerations.

Additional manuals and prints for your Conair equipment may be ordered through the Customer Service or Parts Departments for a nominal fee.

Equipment Guarantee

Conair guarantees the machinery and equipment on this order, for a period as defined in the quotation from date of shipment, against defects in material and workmanship under the normal use and service for which it was recommended (except for parts that are typically replaced after normal usage, such as filters, liner plates, etc.). Conair's guarantee is limited to replacing, at our option, the part or parts determined by us to be defective after examination. The customer assumes the cost of transportation of the part or parts to and from the factory.

Performance Warranty

Conair warrants that this equipment will perform at or above the ratings stated in specific quotations covering the equipment or as detailed in engineering specifications, provided the equipment is applied, installed, operated and maintained in the recommended manner as outlined in our quotation or specifications.

Should performance not meet warranted levels, Conair at its discretion will exercise one of the following options:

- Inspect the equipment and perform alterations or adjustments to satisfy performance claims. (Charges for such inspections and corrections will be waived unless failure to meet warranty is due to misapplication, improper installation, poor maintenance practices or improper operation.)
- Replace the original equipment with other Conair equipment that will meet original performance claims at no extra cost to the customer.
- Refund the invoiced cost to the customer. Credit is subject to prior notice by the customer at which time a Return Goods Authorization Number (RGA) will be issued by Conair's Service Department. Returned equipment must be well crated and in proper operating condition, including all parts. Returns must be prepaid.

Purchaser must notify Conair in writing of any claim and provide a customer receipt and other evidence that a claim is being made.

Warranty Limitations

Except for the Equipment Guarantee and Performance Warranty stated above, Conair disclaims all other warranties with respect to the equipment, express or implied, arising by operation of law, course of dealing, usage of trade or otherwise, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.