

USERGUIDE

# SYSTEM CONFIGURATION

## for S900II robots

### Software Version 1.0



**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This manual is out-of-date and is provided only for its technical information, data and capacities.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](mailto: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containing warnings, information, precautions, and procedures that may be more adequate than those contained in this out-of-date manual.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



Sepro robotique innovations



What to do ?



Document evolutions



Handy hints



Example



Software innovation

## I – MEMORY

### I – 1. Accessing the memory

After accessing "Memory Management" by pressing [Memo\_M] (programming menu), pressing the [M\_Read] key gives access to the read (or modification) function of the user and system RAM or EEPROM memory (at the address of the memory box by default if necessary).

The address of the area at 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).



: transfer buffer PRG E17.



: robot serial number in RAM.



: RAM access password.



: 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 :

[Memo\_M] -> [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).

\* **The function keys F1 to F5 :**

- ▶ **[Address]**      to change the address.
- ▶ **[Modif]**      to change the contents of the memory area displayed (word).
- ▶ **[Search]**      to search for a particular word (e.g. : FA1B)
- ▶ **[Print]**      to print the memory contents from the displayed address (in order to search for the incorrect instructions which will be printed as ???).
- ▶ **[StopPr]**      to stop sending the memory contents to the printer.

---

Note : To access the modification function, a password is necessary which remains valid as long as the user does not exit the “M\_Read” procedure. Certain critical system areas cannot be read and all requests to modify them will be rejected.

By default, the value given after modification request is 0 x FFFF (useful to delete words in the memory).

As for the other functions, the EXIT key is used to abandon a request or to exit the procedure.

**I – 2. Memory areas**

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

| Address in Hexadecimal | Contents  |
|------------------------|---|
| 00000                  | Variables used by Philips (BOOT)                                      |
| 027FF<br>02800         |   |
| 0A4FF<br>0A500         | “Fixed” SEPRO variables, see table below for details of the variables |
| 0B2FF<br>0B300         | SEPRO parameters in RAM   |
| 2A6FF<br>2A700         | PRG storage area (128 K × 8)  |
| 37FFF<br>38000         | SEPRO variables / work tables   |
| 57FFF<br>58000         | Temporary transfer area (128 K x 8)                                   |
| 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 addressing in 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 [ RsMEM ] (on the size provided for in the options)

The parameters are stored in FLASHROM 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 FLASHROM at the address 0xF10E1200. Its length is fixed at 4590 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 Flashrom (1 M x 8) F10 00000 to F10 FFFFF**

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

| Block number                       | Address in Hexadecimal | Contents                       |
|------------------------------------|------------------------|--------------------------------|
| 7th block<br>Messages              | F10 C0000              | Messages in language 1         |
|                                    | F10 CEBEF              |                                |
|                                    | F10 CEBF0              | Messages in language 2         |
|                                    | F10 DD7DF              |                                |
|                                    | F10 DD7E0              | Font robot 1                   |
|                                    | F10 DE7EF              |                                |
|                                    | F10 DE7F0              | Font robot 2                   |
|                                    | F10 DF7FF              |                                |
|                                    | F10 DF800              | Code converter table IMM 1     |
|                                    | F10 DF9FF              |                                |
|                                    | F10 DFA00              | Code converter table IMM 2     |
|                                    | F10 DFBFF              |                                |
|                                    | F10 DFC00              | Code converter table Printer 1 |
|                                    | F10 DFDFF              |                                |
|                                    | F10 DFE00              | Code converter table Printer 2 |
| F10 DFFFF                          |                        |                                |
| 8th block<br>Parameters<br>and SAP | F10 E0000              | SEPRO parameters               |
|                                    | F10 E0DFF              |                                |
|                                    | F10 E1200              | SAP messages                   |
|                                    | F10 E2256              |                                |
|                                    | F10 E2400              | Reserved for SEPRO             |
| F10 FFFFF                          |                        |                                |

**I – 3. Specific information**

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

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

|      | 15 | 0  |  |
|------|----|----|--|
| B2A0 | 00 | 00 | Password to access edition (...)   |
| B2A2 | 00 | 00 |  |
| B2A4 | 00 | 00 | Password to access parameters (...)  |
| B2A6 | 04 | D2 |  |
| B2A8 | 00 | 00 | Password to access maintenance (...)   |
| B2AA | 00 | 00 |  |
| B2AC | 00 | 00 | Password to block the modes (...)  |
| B2AE | 00 | 00 |  |
| B2B0 | 00 | 00 | Password to block the selection of the PRG N° to be executed (...)   |
| B2B4 | 00 | 00 |  |
| ⋮    |    |    |  |
| B2E0 |    |    | Operating time.  |
| B2E2 |    |    |  |
| B2E4 |    |    | Operating time in automatic.   |
| B2E6 |    |    |  |
| B2E8 | 00 | 00 | Robot serial number :  |
| B2EA | 04 | 00 | E.g. 1024  |
| B2EC | 00 | 35 | Robot type :   |
| B2EE | 73 | 98 | E.g. 350 BB (000) -> 3503000-D -> 357398-H   |
| ⋮    |    |    | <div style="display: flex; justify-content: space-around; margin-top: 10px;"> <span>Model</span> <span>Type</span> <span>Specific</span> </div> <div style="margin-left: 40px; margin-top: 10px;">             0 BX<br/>             1 BY<br/>             2 BZ<br/>             3 BB<br/>             4 BC<br/>             5 AX<br/>             6 AY<br/>             7 AZ           </div> |



## II – INSTRUCTION CODES

### II – 1. Part programs

| Type of Instruction  | Display            | Codop (hexadecimal)   | Examples   |
|----------------------|--------------------|---|--|
| <b>ACTION</b>        | ACT 00 (to 99) *   | A000 [oper. 16 bits]<br>↓<br>Action No.   | A000000C = ACT12                                       |
| <b>OUTPUT</b>        | OUT 000 (to 255) * | A001 [oper. 16 bits]<br>↓<br>Output No.   | A0010050 = OUT080                                      |
| <b>INPUT Normal</b>  | IN 000 (to 255)    | A002 [oper. 16 bits]<br>↓<br>Input No.  | A002000A = IN010                                       |
| <b>INPUT Reverse</b> | IN/000 (to 255)    | A003 [oper. 16 bits]<br>↓<br>Input No.  | A0030020 = IN/032                                      |
| <b>TIMER</b>         | TIME 001 to 999    | A004[oper.4bits]0[oper.11bits]<br>↓                                       ↓<br>SAP marker    Value<br>No.            in 1/10s | A004000A = TIME010<br>A004300A = TIME010<br>Marker P03 |
|                      | TIME W_00 à 15     | A004 0000 1 [oper.11bits]<br>↓<br>Word No.  | A004080A = TIMEW10<br>A004080F = TIMEW15               |
| <b>BIT</b>           | BIT 000 (to 127)   | A005 [oper. 16 bits]<br>↓<br>Bit No.  | A0050063 = BIT 99                                      |
|                      | / BIT 000 (to 127) | A006 [oper. 16 bits]  | A006007D = BIT 127                                     |

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

| Type of Instruction  | Display  | Codop (hexadecimal)   | Examples  |
|--|--|---|---|
| <b><u>FUNCTIONS (FUNC)</u></b>                                   |  |   |   |
| <b>SPEED<br/>in % of the<br/>parametered speed</b>               | VEL.X 001 to 100<br>VEL.Y 001 to 100<br>VEL.Z 001 to 100<br>VEL.B 001 to 100<br>VEL.C 001 to 100<br><br>VEL.X WW_*nn<br>VEL.Y WW_*nn<br>VEL.Z WW_*nn<br>VEL.B WW_*nn<br>VEL.C WW_*nn<br>*(nn = 00 to 55<br>and 66 to 67) | B000[oper.4bits][oper.12bits]<br>B001[oper.4bits][oper.12bits]<br>B002[oper.4bits][oper.12bits]<br>B003[oper.4bits][oper.12bits]<br>B004[oper.4bits][oper.12bits]<br><br>SAP marker      Value in<br>N°                      1/10s<br><br><br>B050 0000 [oper.12bits]<br>B051 0000 [oper.12bits]<br>B052 0000 [oper.12bits]<br>B053 0000 [oper.12bits]<br>B054 0000 [oper.12bits]<br><br>Word No. | B0000062 = VEL.X 098<br>B001000A = VEL.Y 010<br>B0020012 = VEL.Z 018<br>B0030064 = VEL.B 100<br>B004A032 = VEL.C 050<br>Marker P10<br><br><br>B0500042 = VEL.X <sup>ww066</sup><br>B0510043 = VEL.Y <sup>ww067</sup><br>B0520042 = VEL.Z <sup>ww066</sup><br>B0530042 = VEL.B <sup>ww066</sup><br>B0540043 = VEL.C <sup>ww067</sup> |
| <b>ACCELERATION<br/>in % of the parametered<br/>acceleration</b> | ACC.X 001 to 100<br>ACC.Y 001 to 100<br>ACC.Z 001 to 100<br>ACC.B 001 to 100<br>ACC.C 001 to 100   | B010 [oper. 16 bits]<br>B011 [oper. 16 bits]<br>B012 [oper. 16 bits]<br>B013 [oper. 16 bits]<br>B014 [oper. 16 bits]<br>Value in %  | B010000F = ACC.X 015<br>B0110064 = ACC.Y 100<br>B0120044 = ACC.Z 068<br>B0130005 = ACC.B 005<br>B0140032 = ACC.C 050  |
| <b>Master MOVEMENT</b>   | MASTER.X<br>MASTER.Y<br>MASTER.Z<br>MASTER.B<br>MASTER.C   | B030<br>B031<br>B032<br>B033<br>B034  |   |
| <b>IMPRECISION</b>   | IMP.X<br>IMP.Y<br>IMP.Z<br>IMP.B<br>IMP.C  | B040<br>B041<br>B042<br>B043<br>B044  |   |

| Type of Instruction  | Display   | Codop (hexadecimal)   | Examples   |
|--|---|---|--|
| <p><b><u>MOTORIZED MOTIONS</u></b></p> <p><b>SLOW APPROACH</b><br/>in % of the maximum parametered speed</p>                       | <p>SLA.X 001 to 100<br/>SLA.Y 001 to 100<br/>SLA.Z 001 to 100<br/>SLA.B 001 to 100<br/>SLA.C 001 to 100</p>   | <p>B020 [oper. 16 bits]<br/>B021 [oper. 16 bits]<br/>B022 [oper. 16 bits]<br/>B023 [oper. 16 bits]<br/>B024 [oper. 16 bits]</p> <p style="text-align: center;">▼<br/>Value in %</p>   | <p>B0200026 = SLA.X 026<br/>B0210034 = SLA.Y 034<br/>B0220090 = SLA.Z 090<br/>B0230100 = SLA.B 100<br/>B0240010 = SLA.C 010</p>  |
| <p><b><u>LINEAR ABSOLUTE</u></b><br/>(Numerical operands)</p> <p><b>STACKING</b></p> <p><b>RELATIVE</b></p> <p><b>CHECKING</b></p> | <p>X.ABS_L distance<br/>Y.ABS_L distance<br/>Z.ABS_L distance<br/>B.ABS_L distance<br/>C.ABS_L distance</p> <p>X.STK_L distance<br/>Y.STK_L distance<br/>Z.STK_L distance<br/>B.STK_L distance<br/>C.STK_L distance</p> <p>X.REL_L distance<br/>Y.REL_L distance<br/>Z.REL_L distance<br/>B.REL_L distance<br/>C.REL_L distance</p> <p>X.CTL_L distance<br/>Y.CTL_L distance<br/>Z.CTL_L distance<br/>B.CTL_L distance<br/>C.CTL_L distance</p> | <p>C000[oper.8bits][oper.24bits]<br/>C001[oper.8bits][oper.24bits]<br/>C002[oper.8bits][oper.24bits]<br/>C003[oper.8bits][oper.24bits]<br/>C004[oper.8bits][oper.24bits]</p> <p>C010[oper.8bits][oper.24bits]<br/>C011[oper.8bits][oper.24bits]<br/>C012[oper.8bits][oper.24bits]<br/>C053<br/>C054</p> <p>C020[oper.8bits][oper.24bits]<br/>C021[oper.8bits][oper.24bits]<br/>C022[oper.8bits][oper.24bits]<br/>C023[oper.8bits][oper.24bits]<br/>C024[oper.8bits][oper.24bits]</p> <p>C030[oper.8bits][oper.24bits]<br/>C031[oper.8bits][oper.24bits]<br/>C032[oper.8bits][oper.24bits]<br/>C033[oper.8bits][oper.24bits]<br/>C034[oper.8bits][oper.24bits]</p> <p style="text-align: center;">▼                      ▼<br/>SAP marker No.      Distance in 1/10 mm</p> | <p>C0000000664=X.ABS.L00163.6<br/>C001000F423F=Y.ABS.L99999.9<br/>C00200000320=Z.ABS.L00080.0<br/>C0030000003F=B.ABS.L00006.3<br/>C0040000050C=C.ABS.L00150.0</p> <p>C0100008ACF=X.STK.L03453.5<br/>C01100030DE3=Y.STK.L20016.3<br/>C01200000159=Z.STK.L00034.5<br/>Reserved for general STKs<br/>Absolute distances from the header</p> <p>C020800000A0=X.REL.L-0016.0<br/>C021000000A0=Y.REL.L-0016.0<br/>C0228001869F=Z.REL.L-9999.9<br/>C02300002706=B.REL.L+0999.9<br/>C0240000000A=C.REL.L+0001.0</p> <p>C03000000664=X.CTL.L00163.6<br/>C031000F423F=Y.CTL.L9999.9<br/>C03200000320=Z.CTL.L00080.0<br/>C0330000003F=B.CTL.L00006.3<br/>C0340500050C=C.CTL.L00150.0</p> <p style="text-align: right;">Marker P05</p> |
| <p><b><u>ROTATING ABSOLUTE</u></b><br/>(Numerical operands)</p> <p><b>STACKING</b></p> <p><b>RELATIVE</b></p>                      | <p>X.ABS_R Angle<br/>Y.ABS_R Angle<br/>Z.ABS_R Angle<br/>B.ABS_R Angle<br/>C.ABS_R Angle</p> <p>X.STK_R Angle<br/>Y.STK_R Angle<br/>Z.STK_R Angle</p> <p>X.REL_R Angle<br/>Y.REL_R Angle<br/>Z.REL_R Angle<br/>B.REL_R Angle<br/>C.REL_R Angle</p>  | <p>C100[oper.8bits][oper.24bits]<br/>C101[oper.8bits][oper.24bits]<br/>C102[oper.8bits][oper.24bits]<br/>C103[oper.8bits][oper.24bits]<br/>C104[oper.8bits][oper.24bits]</p> <p>C110[oper.8bits][oper.24bits]<br/>C111[oper.8bits][oper.24bits]<br/>C112[oper.8bits][oper.24bits]</p> <p>C120[oper.8bits][oper.24bits]<br/>C121[oper.8bits][oper.24bits]<br/>C122[oper.8bits][oper.24bits]<br/>C123[oper.8bits][oper.24bits]<br/>C124[oper.8bits][oper.24bits]</p>  | <p>C1000000664=X.ABS.R00163.6<br/>C101000005DC=Y.ABS.R00150.0<br/>C10200000320=Z.ABS.R00080.0<br/>C1030000003F=B.ABS.R00006.3<br/>C10400000159=C.ABS.R00034.5</p> <p>C1100008ACF=X.STK.R03453.5<br/>C11100030DE3=Y.STK.R20016.3<br/>C11200000159=Z.STK.R00034.5</p> <p>C12000000384=X.REL.R+90.0<br/>C12180000320=Y.REL.R-90.0<br/>C12200000320=Z.REL.R+80.0<br/>C12380000159=B.REL.R-34.5<br/>C1240000003F=C.REL.R+06.3</p>   |

| Type of Instruction                      | Display  | Codop (hexadecimal)  | Examples  |
|--|--|--|---|
| <b>CHECKING</b>                          | X.CTL_R Angle<br>Y.CTL_R Angle<br>Z.CTL_R Angle<br>B.CTL_R Angle<br>C.CTL_R Angle      | C130[oper.8bits][oper.24bits]<br>C131[oper.8bits][oper.24bits]<br>C132[oper.8bits][oper.24bits]<br>C133[oper.8bits][oper.24bits]<br>C134[oper.8bits][oper.24bits]<br>↓   ↓<br>SAP Marker No.           Angle in<br>1/10 deg. | C1300000664=X.CTL.R00163.6<br>C131000F423F=Y.CTL.R9999.9<br>C13200000320=Z.CTL.R00080.0<br>C1330000003F=B.CTL.R00006.3<br>C1340000050C=C.CTL.R00150.0 |
| <b>TEACHING</b>                          | □ □ □ Teach<br>↓<br>Previous instruction   | IC   AAAAAA<br>↓   ↓<br>Instruction code   SAP marker No.  | C01000AAAAAA=X.STK.LTeach<br>C10200AAAAAA=Z.ABS.RTeach  |
| <b><u>MOTORIZED MOTIONS</u> (cont'd)</b> |  |  |   |
| <b><u>LINEAR</u></b>                     |  |  |   |
| <b>ABSOLUTE</b><br>(Words)               | X.ABS_L WW *nn<br>Y.ABS_L WW *nn<br>Z.ABS_L WW *nn<br>B.ABS_L WW *nn<br>C.ABS_L WW *nn | C200 [oper. 16 bits]<br>C201 [oper. 16 bits]<br>C202 [oper. 16 bits]<br>C203 [oper. 16 bits]<br>C204 [oper. 16 bits]   | C200000A = X.ABS.L WW10   |
| <b>STACKING</b>                          | X.STK_L WW *nn<br>Y.STK_L WW *nn<br>Z.STK_L WW *nn                                     | C210 [oper. 16 bits]<br>C211 [oper. 16 bits]<br>C212 [oper. 16 bits]   | C210000B = X.STK.L WW11   |
| <b>RELATIVE</b>                          | X.REL_L WW *nn<br>Y.REL_L WW *nn<br>Z.REL_L WW *nn<br>B.REL_L WW *nn<br>C.REL_L WW *nn | C220 [oper. 16 bits]<br>C221 [oper. 16 bits]<br>C222 [oper. 16 bits]<br>C223 [oper. 16 bits]<br>C224 [oper. 16 bits]   | C2200041 = X.REL.L WW65   |
| <b>CHECKING</b>                          | X.CTL_L WW *nn<br>Y.CTL_L WW *nn<br>Z.CTL_L WW *nn<br>B.CTL_L WW *nn<br>C.CTL_L WW *nn | C230 [oper. 16 bits]<br>C231 [oper. 16 bits]<br>C232 [oper. 16 bits]<br>C233 [oper. 16 bits]<br>C234 [oper. 16 bits]   | C2300010 = X.CTL.L WW16   |
| <b><u>ROTATING</u></b>                   |  |  |   |
| <b>ABSOLUTE</b><br>(Words)               | X.ABS_R WW *nn<br>Y.ABS_R WW *nn<br>Z.ABS_R WW *nn<br>B.ABS_R WW *nn<br>C.ABS_R WW *nn | C300 [oper. 16 bits]<br>C301 [oper. 16 bits]<br>C302 [oper. 16 bits]<br>C303 [oper. 16 bits]<br>C304 [oper. 16 bits]   | C300000A = X.ABS.R WW10   |
| <b>STACKING</b>                          | X.STK_R WW *nn<br>Y.STK_R WW *nn<br>Z.STK_R WW *nn                                     | C310 [oper. 16 bits]<br>C311 [oper. 16 bits]<br>C312 [oper. 16 bits]   | C3100020 = X.STK.R WW32   |
|  | *(nn = 00 to 55 and 64 to 65)  |  |   |

| Type of Instruction                                    | Display   | Codop (hexadecimal)   | Examples   |
|--|---|---|--|
| <b>RELATIVE</b><br><br><br><br><br><br><b>CHECKING</b> | X.REL_R WW *nn<br>Y.REL_R WW *nn<br>Z.REL_R WW *nn<br>B.REL_R WW *nn<br>C.REL_R WW *nn<br><br>X.CTL_R WW *nn<br>Y.CTL_R WW *nn<br>Z.CTL_R WW *nn<br>B.CTL_R WW *nn<br>C.CTL_R WW *nn<br>*(nn = 00 to 55 and 64 to 65) | C320 [oper. 16 bits]<br>C321 [oper. 16 bits]<br>C322 [oper. 16 bits]<br>C323 [oper. 16 bits]<br>C324 [oper. 16 bits]<br><br>C330 [oper. 16 bits]<br>C331 [oper. 16 bits]<br>C332 [oper. 16 bits]<br>C333 [oper. 16 bits]<br>C334 [oper. 16 bits]<br>WWORD No. | C3200001 = X.REL.R WW01<br><br><br><br><br><br><br><br><br><br><br>C3300041 = X.CTL.R WW65 |
| <b>FREE</b>  | X. FREE<br>Y. FREE<br>Z. FREE<br>B. FREE<br>C. FREE   | C040<br>C041<br>C042<br>C043<br>C044  |  |
| <b>LINE</b>  | LIN.  | B046  |  |
|  |   |   |  |

| Type of Instruction   | Display   | Codop (hexadecimal)   | Examples |
|---|---|---|----------|
| <p><b><u>LINEAR</u></b></p> <p><b>POS_ANA</b></p> <p><b>POS_NUM</b></p> <p><b>VEL ANA NORMAL</b></p> <p><b>VEL ANA INTEGRAL</b></p> <p><b>VEL NUM NORMAL</b></p> <p><b>VEL NUM INTEGRAL</b></p> | <p>X = POS ANA + distance<br/>Y = POS ANA + distance<br/>Z = POS ANA + distance<br/>B = POS ANA + distance<br/>C = POS ANA + distance</p> <p>X = POS NUM + distance<br/>Y = POS NUM + distance<br/>Z = POS NUM + distance<br/>B = POS NUM + distance<br/>C = POS NUM + distance</p> <p>X = VEL ANA_N + distance<br/>Y = VEL ANA_N + distance<br/>Z = VEL ANA_N + distance<br/>B = VEL ANA_N + distance<br/>C = VEL ANA_N + distance</p> <p>X = VEL ANA_I + distance<br/>Y = VEL ANA_I + distance<br/>Z = VEL ANA_I + distance<br/>B = VEL ANA_I + distance<br/>C = VEL ANA_I + distance</p> <p>X = VEL NUM_N + distance<br/>Y = VEL NUM_N + distance<br/>Z = VEL NUM_N + distance<br/>B = VEL NUM_N + distance<br/>C = VEL NUM_N + distance</p> <p>X = VEL NUM_I + distance<br/>Y = VEL NUM_I + distance<br/>Z = VEL NUM_I + distance<br/>B = VEL NUM_I + distance<br/>C = VEL NUM_I + distance</p> | <p>C060 [oper. 32 bits]<br/>C061 [oper. 32 bits]<br/>C062 [oper. 32 bits]<br/>C063 [oper. 32 bits]<br/>C064 [oper. 32 bits]</p> <p>C070 [oper. 32 bits]<br/>C071 [oper. 32 bits]<br/>C072 [oper. 32 bits]<br/>C073 [oper. 32 bits]<br/>C074 [oper. 32 bits]</p> <p>C080 [oper. 32 bits]<br/>C081 [oper. 32 bits]<br/>C082 [oper. 32 bits]<br/>C083 [oper. 32 bits]<br/>C084 [oper. 32 bits]</p> <p>C090 [oper. 32 bits]<br/>C091 [oper. 32 bits]<br/>C092 [oper. 32 bits]<br/>C093 [oper. 32 bits]<br/>C094 [oper. 32 bits]</p> <p>C0A0 [oper. 32 bits]<br/>C0A1 [oper. 32 bits]<br/>C0A2 [oper. 32 bits]<br/>C0A3 [oper. 32 bits]<br/>C0A4 [oper. 32 bits]</p> <p>C0B0 [oper. 32 bits]<br/>C0B1 [oper. 32 bits]<br/>C0B2 [oper. 32 bits]<br/>C0B3 [oper. 32 bits]<br/>C0B4 [oper. 32 bits]</p> |          |

| Type of Instruction   | Display   | Codop (hexadecimal)   | Examples |
|---|---|---|----------|
| <p><b><u>ROTATING</u></b></p> <p><b>POS_ANA</b></p> <p><b>POS_NUM</b></p> <p><b>VEL ANA NORMAL</b></p> <p><b>VEL ANA INTEGRAL</b></p> <p><b>VEL NUM NORMAL</b></p> <p><b>VEL NUM INTEGRAL</b></p> | <p>X = POS ANA + angle<br/>Y = POS ANA + angle<br/>Z = POS ANA + angle<br/>B = POS ANA + angle<br/>C = POS ANA + angle</p> <p>X = POS NUM + angle<br/>Y = POS NUM + angle<br/>Z = POS NUM + angle<br/>B = POS NUM + angle<br/>C = POS NUM + angle</p> <p>X = VEL ANA_N + angle<br/>Y = VEL ANA_N + angle<br/>Z = VEL ANA_N + angle<br/>B = VEL ANA_N + angle<br/>C = VEL ANA_N + angle</p> <p>X = VEL ANA_I + angle<br/>Y = VEL ANA_I + angle<br/>Z = VEL ANA_I + angle<br/>B = VEL ANA_I + angle<br/>C = VEL ANA_I + angle</p> <p>X = VEL NUM_N + angle<br/>Y = VEL NUM_N + angle<br/>Z = VEL NUM_N + angle<br/>B = VEL NUM_N + angle<br/>C = VEL NUM_N + angle</p> <p>X = VEL NUM_I + angle<br/>Y = VEL NUM_I + angle<br/>Z = VEL NUM_I + angle<br/>B = VEL NUM_I + angle<br/>C = VEL NUM_I + angle</p> | <p>C160 [oper. 32 bits]<br/>C161 [oper. 32 bits]<br/>C162 [oper. 32 bits]<br/>C163 [oper. 32 bits]<br/>C164 [oper. 32 bits]</p> <p>C170 [oper. 32 bits]<br/>C171 [oper. 32 bits]<br/>C172 [oper. 32 bits]<br/>C173 [oper. 32 bits]<br/>C174 [oper. 32 bits]</p> <p>C180 [oper. 32 bits]<br/>C181 [oper. 32 bits]<br/>C182 [oper. 32 bits]<br/>C183 [oper. 32 bits]<br/>C184 [oper. 32 bits]</p> <p>C190 [oper. 32 bits]<br/>C191 [oper. 32 bits]<br/>C192 [oper. 32 bits]<br/>C193 [oper. 32 bits]<br/>C194 [oper. 32 bits]</p> <p>C1A0 [oper. 32 bits]<br/>C1A1 [oper. 32 bits]<br/>C1A2 [oper. 32 bits]<br/>C1A3 [oper. 32 bits]<br/>C1A4 [oper. 32 bits]</p> <p>C1B0 [oper. 32 bits]<br/>C1B1 [oper. 32 bits]<br/>C1B2 [oper. 32 bits]<br/>C1B3 [oper. 32 bits]<br/>C1B4 [oper. 32 bits]</p> |          |

| Type of Instruction                  | Display                     | Codop (hexadecimal)  | Examples   |
|--------------------------------------|-----------------------------|----------------------|--|
| <b><u>TEST, CONDITIONS</u></b>       |                             |                      |  |
| <b>. 1 Operand</b>                   |                             |                      |  |
| <b>on Bit</b>                        | IF BIT 000 (to 127)         | D000 [oper. 16 bits] |  |
| <b>on Output</b>                     | IF/BIT 000 (to 127)         | D010 [oper. 16 bits] |  |
| <b>on Input</b>                      | IF OUT 000 (to 255)         | D001 [oper. 16 bits] |  |
| <b>on Timer</b>                      | IF/OUT 000 (to 255)         | D011 [oper. 16 bits] |  |
|                                      | IF IN/000 (to 255)          | D002 [oper. 16 bits] |  |
|                                      | IF IN 000 (to 255)          | D003 [oper. 16 bits] |  |
|                                      | IF/IN 000 (to 255)          | D013 [oper. 16 bits] |  |
|                                      | IF TIM 00 (to 15)           | D004 [oper. 16 bits] |  |
|                                      | IF/TIM 00 (to 15)           | D014 [oper. 16 bits] |  |
|                                      |                             | ↓<br>Operand No.     |  |
| <b>. 2 Operands</b>                  |                             |                      |  |
| * on Word (16 bits)<br>→ 1st Operand | <b>IF WRD 000</b> (to 4095) | D300 [oper. 16 bits] |  |
|                                      | IF/WRD 000 (to 4095)        | D310 [oper. 16 bits] |  |
| <b>with decimal value</b>            | = 0000 (to 9999)            | D400 [oper. 16 bits] | <i>Note</i> : If the decimal value cannot exceed 9,999, the hexadecimal value goes up to 65,535. |
|                                      | >= 0000 (to 9999)           | D401 [oper. 16 bits] |  |
|                                      | <= 0000 (to 9999)           | D402 [oper. 16 bits] |  |
|                                      | AND 0000 (to 9999)          | D403 [oper. 16 bits] |  |
| <b>with hexadecimal value</b>        | = 0000 (to FFFF)            | D410 [oper. 16 bits] |  |
|                                      | >= 0000 (to FFFF)           | D411 [oper. 16 bits] |  |
|                                      | <= 0000 (to FFFF)           | D412 [oper. 16 bits] |  |
|                                      | AND 0000 (to FFFF)          | D413 [oper. 16 bits] |  |
| <b>with Counter</b>                  | = CNT 00 (to 15)            | D420 [oper. 16 bits] |  |
|                                      | >= CNT 00 (to 15)           | D421 [oper. 16 bits] |  |
|                                      | <=CNT 00 (to 15)            | D422 [oper. 16 bits] |  |
|                                      | AND CNT 00 (to 15)          | D423 [oper. 16 bits] |  |
| <b>with Inputs (modulo 16)</b>       | =IN 000 (to 112)            | D430 [oper. 16 bits] |  |
|                                      | >=IN 000 (to 112)           | D431 [oper. 16 bits] |  |
|                                      | <=IN 000 (to 112)           | D432 [oper. 16 bits] |  |
|                                      | AND IN 000 (to 112)         | D433 [oper. 16 bits] |  |
| <b>with Word (16 bits)</b>           | = WRD 0000 (to 4095)        | D440 [oper. 16 bits] |  |
|                                      | >= WRD 0000 (to 4095)       | D441 [oper. 16 bits] |  |
|                                      | <= WRD 0000 (to 4095)       | D442 [oper. 16 bits] |  |
|                                      | AND WRD 0000(to 4095)       | D443 [oper. 16 bits] |  |



| Type of Instruction   | Display   | Codop (hexadecimal)  | Examples   |
|---|---|--|--|
| <p>* on WWord (32 bits)<br/>→ 1st Operand</p> <p><b>with decimal value</b></p> <p><b>with hexadecimal value</b></p> <p><b>with Counter</b></p> <p><b>with Inputs (modulo 16)</b></p> <p><b>with Word (16 bits)</b></p> <p><b>with WWord (32 bits)</b></p> | <p><b>IF WWRD</b> 000 (to 127)<br/><b>IF/WWRD</b> 000 (to 127)</p> <p>= 00000000 (to 09999999)<br/>&gt; = 00000000 (to 09999999)<br/>&lt; = 00000000 (to 09999999)<br/><b>AND</b> 00000000 (to 09999999)</p> <p>= 00000000 (to FFFFFFFF)<br/>&gt; = 00000000 (to FFFFFFFF)<br/>&lt; = 00000000 (to FFFFFFFF)<br/><b>AND</b>00000000 (to FFFFFFFF)</p> <p>= CNT 00 (to 15)<br/>&gt; = CNT 00 (to 15)<br/>&lt; = CNT 00 (to 15)<br/><b>AND</b> CNT 00 (to 15)</p> <p>= IN 000 (to 112)<br/>&gt; = IN 000 (to 112)<br/>&lt; = IN 000 (to 112)<br/><b>AND</b> IN 000 (to 112)</p> <p>= WRD 0000 (to 4095)<br/>&gt; = WRD 0000 (to 4095)<br/>&lt; = WRD 0000 (to 4095)<br/><b>AND</b> WRD 0000(to 4095)</p> <p>= WWRD 000 (to 127)<br/>&gt; = WWRD 000 (to 127)<br/>&lt; = WWRD 000 (to 127)<br/><b>AND</b> WWRD 000(to 127)</p> | <p>D320 [oper. 16 bits]<br/>D330 [oper. 16 bits]</p> <p>D500 [oper. 32 bits]<br/>D501 [oper. 32 bits]<br/>D502 [oper. 32 bits]<br/>D503 [oper. 32 bits]</p> <p>D510 [oper. 32 bits]<br/>D511 [oper. 32 bits]<br/>D512 [oper. 32 bits]<br/>D513 [oper. 32 bits]</p> <p>D520 [oper. 16 bits]<br/>D521 [oper. 16 bits]<br/>D522 [oper. 16 bits]<br/>D523 [oper. 16 bits]</p> <p>D530 [oper. 16 bits]<br/>D531 [oper. 16 bits]<br/>D532 [oper. 16 bits]<br/>D533 [oper. 16 bits]</p> <p>D540 [oper. 16 bits]<br/>D541 [oper. 16 bits]<br/>D542 [oper. 16 bits]<br/>D543 [oper. 16 bits]</p> <p>D550 [oper. 16 bits]<br/>D551 [oper. 16 bits]<br/>D552 [oper. 16 bits]<br/>D553 [oper. 16 bits]</p> | <p><i>Note</i> : If the decimal value cannot exceed 9,999,999, the hexadecimal value goes up to 4,294,967,295.</p> |
| <p>* on Counter<br/>→ 1st Operand</p> <p><b>with decimal value</b></p> <p><b>with hexadecimal value</b></p> <p><b>with Counter</b></p>  | <p><b>IF CNT</b> 00 (to 15)<br/><b>IF/CNT</b> 00 (to 15)</p> <p>= 0000 (to 9999)<br/>&gt; = 0000 (to 9999)<br/>&lt; = 0000 (to 9999)<br/><b>AND</b> 0000 (to 9999)</p> <p>= 0000 (to FFFF)<br/>&gt; = 0000 (to FFFF)<br/>&lt; = 0000 (to FFFF)<br/><b>AND</b> 0000 (to FFFF)</p> <p>= CNT 00 (to 15)<br/>&gt; = CNT 00 (to 15)<br/>&lt; = CNT 00 (to 15)<br/><b>AND</b> CNT 00 (to 15)</p>  | <p>D340 [oper. 16 bits]<br/>D350 [oper. 16 bits]</p> <p>D900 [oper. 16 bits]<br/>D901 [oper. 16 bits]<br/>D902 [oper. 16 bits]<br/>D903 [oper. 16 bits]</p> <p>D910 [oper. 16 bits]<br/>D911 [oper. 16 bits]<br/>D912 [oper. 16 bits]<br/>D913 [oper. 16 bits]</p> <p>D920 [oper. 16 bits]<br/>D921 [oper. 16 bits]<br/>D922 [oper. 16 bits]<br/>D923 [oper. 16 bits]</p>  |  |

| Type of Instruction  | Display  | Codop (hexadecimal)   | Examples |
|--|--|---|----------|
| <p><b>with Inputs (modulo 16)</b></p> <p><b>with Word (16 bits)</b></p>  | <p>= IN 000 (to 112)<br/>&gt; = IN 000 (to 112)<br/>&lt; = IN 000 (to 112)<br/>AND IN 000 (to 112)</p> <p>= WRD 0000 (to 4095)<br/>&gt; = WRD 0000 (to 4095)<br/>&lt; = WRD 0000 (to 4095)<br/>AND WRD 0000(to 4095)</p>   | <p>D930 [oper. 16 bits]<br/>D931 [oper. 16 bits]<br/>D932 [oper. 16 bits]<br/>D933 [oper. 16 bits]</p> <p>D940 [oper. 16 bits]<br/>D941 [oper. 16 bits]<br/>D942 [oper. 16 bits]<br/>D943 [oper. 16 bits]</p>   |          |
| <p><b><u>INITIALIZATION</u></b></p> <p><b>. 1 Operand</b></p> <p>* on Bit → 1<br/>on Bit → 0</p> <p>* on Output → 1<br/>on Output → 0</p> <p>* on Word → 0</p> <p>* on WWord → 0</p> <p>* on Counter → 0</p> | <p>SET.BIT 032 (to 127)<br/>RST.BIT 032 (to 127)</p> <p>SET.OUT 000 (to 127)<br/>RST.OUT 000 (to 127)</p> <p>RST.WRD 0000 (to 4095)</p> <p>RST.WWRD 00 (to 63)</p> <p>RST.CNT 0000 (to 0015)<br/>RST.CNT 0041 (to 9980)</p>  | <p>D015 [oper. 16 bits]<br/>D017 [oper. 16 bits]</p> <p>D016 [oper. 16 bits]<br/>D018 [oper. 16 bits]</p> <p>D019 [oper. 16 bits]<br/>Variable number<br/>D01D [oper. 16 bits]<br/>Variable number<br/>D01A 00 [oper. 8 bits]<br/>Counter number<br/>D01A[oper. 8 bits] [oper. 8 bits]<br/>PRG No. SP No.</p>   |          |
| <p><b>. 2 Operands</b></p> <p>* on Word (16 bits)<br/>→ 1st Operand</p> <p><b>with decimal value</b></p> <p><b>with hexadecimal value</b></p>  | <p><b>SET.WRD 0000 (to 4095)</b></p> <p>= 0000 (to 9999)<br/>+ 0000 (to 9999)<br/>– 0000 (to 9999)<br/>x 0000 (to 9999)<br/>/ 0000 (to 9999)<br/>AND 0000 (to 9999)<br/>OR 0000 (to 9999)</p> <p>= 0000 (to FFFF)<br/>+ 0000 (to FFFF)<br/>– 0000 (to FFFF)<br/>x 0000 (to FFFF)<br/>/ 0000 (to FFFF)<br/>AND 0000 (to FFFF)<br/>OR 0000 (to FFFF)</p> | <p>D600 [oper. 16 bits]</p> <p>D700 [oper. 16 bits]<br/>D701 [oper. 16 bits]<br/>D702 [oper. 16 bits]<br/>D703 [oper. 16 bits]<br/>D704 [oper. 16 bits]<br/>D705 [oper. 16 bits]<br/>D706 [oper. 16 bits]</p> <p>D710 [oper. 16 bits]<br/>D711 [oper. 16 bits]<br/>D712 [oper. 16 bits]<br/>D713 [oper. 16 bits]<br/>D714 [oper. 16 bits]<br/>D715 [oper. 16 bits]<br/>D716 [oper. 16 bits]</p> |          |

| Type of Instruction  | Display   | Codop (hexadecimal)  | Examples |
|--|---|--|----------|
| <b>with Counter</b>  | = CNT 00 (to 15)<br>+ CNT 00 (to 15)<br>– CNT 00 (to 15)<br>x CNT 00 (to 15)<br>/ CNT00 (to 15)<br>AND CNT 00 (to 15)<br>OR CNT 00 (to 15)  | D720 [oper. 16 bits]<br>D721 [oper. 16 bits]<br>D722 [oper. 16 bits]<br>D723 [oper. 16 bits]<br>D724 [oper. 16 bits]<br>D725 [oper. 16 bits]<br>D726 [oper. 16 bits]                         |          |
| <b>with Inputs (modulo 16)</b>                                     | = IN 000 (to 112)<br>+ IN 000 (to 112)<br>– IN 000 (to 112)<br>x IN 000 (to 112)<br>/ IN 000 (to 112)<br>AND IN 000 (to 112)<br>OR IN 000 (to 112)  | D730 [oper. 16 bits]<br>D731 [oper. 16 bits]<br>D732 [oper. 16 bits]<br>D733 [oper. 16 bits]<br>D734 [oper. 16 bits]<br>D735 [oper. 16 bits]<br>D736 [oper. 16 bits]                         |          |
| <b>with Word (16 bits)</b>   | = WRD 0000 (to 4095)<br>+ WRD 0000 (to 4095)<br>– WRD 0000 (to 4095)<br>x WRD 0000 (to 4095)<br>/ WRD 0000 (to 4095)<br>AND WRD 0000 (to 4095)<br>OR WRD 0000 (to 4095)   | D740 [oper. 16 bits]<br>D741 [oper. 16 bits]<br>D742 [oper. 16 bits]<br>D743 [oper. 16 bits]<br>D744 [oper. 16 bits]<br>D745 [oper. 16 bits]<br>D746 [oper. 16 bits]                         |          |
| * on WWord (32 bits)<br>→ 1st Operand<br><b>with decimal value</b> | <b>SET.WWRD 000</b> (to 127)<br>= 00000000 (to 09999999)<br>+ 00000000 (to 09999999)<br>– 00000000 (to 09999999)<br>x 00000000 (to 09999999)<br>/ 00000000 (to 09999999)<br>AND 00000000 (to 09999999)<br>OR 00000000 (to 09999999) | D620 [oper. 16 bits]<br>D800 [oper. 32 bits]<br>D801 [oper. 32 bits]<br>D802 [oper. 32 bits]<br>D803 [oper. 32 bits]<br>D804 [oper. 32 bits]<br>D805 [oper. 32 bits]<br>D806 [oper. 32 bits] |          |
| <b>with hexadecimal value</b>                                      | = 00000000 (to FFFFFFFF)<br>+ 00000000 (to FFFFFFFF)<br>– 00000000 (to FFFFFFFF)<br>x 00000000 (to FFFFFFFF)<br>/ 00000000 (to FFFFFFFF)<br>AND 00000000 (to FFFFFFFF)<br>OR 00000000 (to FFFFFFFF)                                 | D810 [oper. 32 bits]<br>D811 [oper. 32 bits]<br>D812 [oper. 32 bits]<br>D813 [oper. 32 bits]<br>D814 [oper. 32 bits]<br>D815 [oper. 32 bits]<br>D816 [oper. 32 bits]                         |          |
| <b>with Counter</b>  | = CNT 00 (to 15)<br>+ CNT 00 (to 15)<br>– CNT 00 (to 15)<br>x CNT 00 (to 15)<br>/ CNT 00 (to 15)<br>AND CNT 00 (to 15)<br>OR CNT 00 (to 15)   | D820 [oper. 16 bits]<br>D821 [oper. 16 bits]<br>D822 [oper. 16 bits]<br>D823 [oper. 16 bits]<br>D824 [oper. 16 bits]<br>D825 [oper. 16 bits]<br>D826 [oper. 16 bits]                         |          |

| Type of Instruction  | Display   | Codop (hexadecimal)  | Examples                             |
|--|---|--|--------------------------------------|
| <b>with Inputs (modulo 16)</b><br><br>*nn = 00 to 112<br>and 136 to 240  | = IN *nn<br>+ IN *nn<br>– IN *nn<br>x IN *nn<br>/ IN *nn<br>AND IN *nn<br>OR IN *nn   | D830 [oper. 16 bits]<br>D831 [oper. 16 bits]<br>D832 [oper. 16 bits]<br>D833 [oper. 16 bits]<br>D834 [oper. 16 bits]<br>D835 [oper. 16 bits]<br>D836 [oper. 16 bits]   |                                      |
| <b>with Word (16 bits)</b>   | = WRD 0000 (to 4095)<br>+ WRD 0000 (to 4095)<br>– WRD 0000 (to 4095)<br>x WRD 0000 (to 4095)<br>/ WRD 0000 (to 4095)<br>AND WRD 0000 (to 4095)<br>OR WRD 0000 (to 4095)   | D840 [oper. 16 bits]<br>D841 [oper. 16 bits]<br>D842 [oper. 16 bits]<br>D843 [oper. 16 bits]<br>D844 [oper. 16 bits]<br>D845 [oper. 16 bits]<br>D846 [oper. 16 bits]   |                                      |
| <b>with WWord (32 bits)</b><br><br>*nn = 0 to 127  | = WWRD *nn and 200–202<br>+ WWRD *nn<br>– WWRD *nn<br>x WWRD *nn<br>/ WWRD *nn<br>AND WWRD*nn<br>OR WWRD *nn  | D850 [oper. 16 bits]<br>D851 [oper. 16 bits]<br>D852 [oper. 16 bits]<br>D853 [oper. 16 bits]<br>D854 [oper. 16 bits]<br>D855 [oper. 16 bits]<br>D856 [oper. 16 bits]   |                                      |
| * on Counter<br>→ 1st Operand<br><br><b>with decimal value</b><br><br><b>with hexadecimal value</b><br><br><b>with Counter</b> | <b>SET.CNT</b> 0000 (to 0015)<br><b>SET.CNT</b> 0041 (to 9980)<br><br>= 0000 (to 9999)<br>+ 0000 (to 9999)<br>– 0000 (to 9999)<br>x 0000 (to 9999)<br>/ 0000 (to 9999)<br>AND 0000 (to 9999)<br>OR 0000 (to 9999)<br><br>= 0000 (to FFFF)<br>+ 0000 (to FFFF)<br>– 0000 (to FFFF)<br>x 0000 (to FFFF)<br>/ 0000 (to FFFF)<br>AND 0000 (to FFFF)<br>OR 0000 (to FFFF)<br><br>= CNT 00 (to 15)<br>+ CNT 00 (to 15)<br>– CNT 00 (to 15)<br>x CNT 00 (to 15)<br>/ CNT 00 (to 15)<br>AND CNT 00 (to 15)<br>OR CNT 00 (to 15) | D640 [oper. 8 bits]<br>D640 [oper. 8 bits] [oper. 8 bits]<br>PRG No.      SP No.<br>DA00 [oper. 16 bits]<br>DA01 [oper. 16 bits]<br>DA02 [oper. 16 bits]<br>DA03 [oper. 16 bits]<br>DA04 [oper. 16 bits]<br>DA05 [oper. 16 bits]<br>DA06 [oper. 16 bits]<br><br>DA10 [oper. 16 bits]<br>DA11 [oper. 16 bits]<br>DA12 [oper. 16 bits]<br>DA13 [oper. 16 bits]<br>DA14 [oper. 16 bits]<br>DA15 [oper. 16 bits]<br>DA16 [oper. 16 bits]<br><br>D920 [oper. 16 bits]<br>D921 [oper. 16 bits]<br>D922 [oper. 16 bits]<br>D922 [oper. 16 bits]<br>D922 [oper. 16 bits]<br>D923 [oper. 16 bits]<br>D923 [oper. 16 bits] | Standard counter<br>Stacking counter |

| Type of Instruction            | Display  | Codop (hexadecimal)  | Examples                               |
|--------------------------------|--|--|--|
| <b>with Inputs (modulo 16)</b> | = IN 000 (to 112)<br>+ IN 000 (to 112)<br>– IN 000 (to 112)<br>x IN 000 (to 112)<br>/ IN 000 (to 112)<br>AND IN 000 (to 112)<br>OR IN 000 (to 112)                     | DA30 [oper. 16 bits]<br>DA31 [oper. 16 bits]<br>DA32 [oper. 16 bits]<br>DA33 [oper. 16 bits]<br>DA34 [oper. 16 bits]<br>DA35 [oper. 16 bits]<br>DA36 [oper. 16 bits] |  |
| <b>with Word (16 bits)</b>     | = WRD 0000 (to 4095)<br>+ WRD 0000 (to 4095)<br>– WRD0000 (to 4095)<br>x WRD 0000 (to 4095)<br>/ WRD 0000 (to 4095)<br>AND WRD 0000 (to 4095)<br>OR WRD 0000 (to 4095) | DA40 [oper. 16 bits]<br>DA41 [oper. 16 bits]<br>DA42 [oper. 16 bits]<br>DA43 [oper. 16 bits]<br>DA44 [oper. 16 bits]<br>DA45 [oper. 16 bits]<br>DA46 [oper. 16 bits] |  |
| <b>-&gt; + 1</b>               | INC.CNT 0000 (to 0015)   | D01B 00 [oper. 8 bits]   |  |
|                                | INC.CNT 0041 (to 9980)   | D01B[oper. 8 bits] [oper. 8 bits]  | Standard No.<br>↓<br>PRG No.    SP No. |
| <b>-&gt; - 1</b>               | DEC.CNT 0000 (to 0015)   | D01C 00 [oper. 8 bits]   | Standard No.<br>↓<br>PRG No.    SP No. |
|                                | DEC.CNT 0041 (to 9980)   | D01C[oper. 8 bits] [oper. 8 bits]  | Standard No.<br>↓<br>PRG No.    SP No. |

II – 2. PLC programs

| Type of Instruction      | Display   | Codop (hexadecimal)  |
|--------------------------|---|--|
| PROG.PLC xx header (num) | PLC xx  | FC [oper. 16 bits]<br>↓<br>PLC No.   |
| TEST CONDITION           | IF ...  | See part programs  |
| INITIALISATION           | SET ...<br>RST ...<br>INC ...<br>DEC ...                | See part programs  |
| COMPARISON xxxx > = xxxx | CMP 0000 (to 0015) VAL 0000 (to FFFF)<br>0000 (to 0015) | D020 [oper. 16 bits] [oper. 16 bits]<br>↓ ↓<br>Counter No. Value           |
| TIMER xx VALUE xxxx      | TIMER 00 (to 15) VAL 0000 (to 9999)                     | D021 [oper. 16 bits] [oper. 16 bits]<br>↓ ↓<br>Timer No. Pre-selection No. |
| AND FUNCTION on BIT      | AND BIT 000 (to 127)                                    | D022 [oper. 16 bits]   |
| AND FUNCTION on OUTPUT   | AND OUT 000 (to 127)                                    | D023 [oper. 16 bits]   |
| AND FUNCTION on BIT      | OR BIT 000 (to 127)                                     | D024 [oper. 16 bits]   |
| OR FUNCTION on OUTPUT    | OR OUT 000 (to 127)                                     | D025 [oper. 16 bits]<br>↓<br>Variables No.                                 |
| END OF PROGRAM           | END   | F5 [oper. 16 bits]<br>↓<br>PLC No.   |

## 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

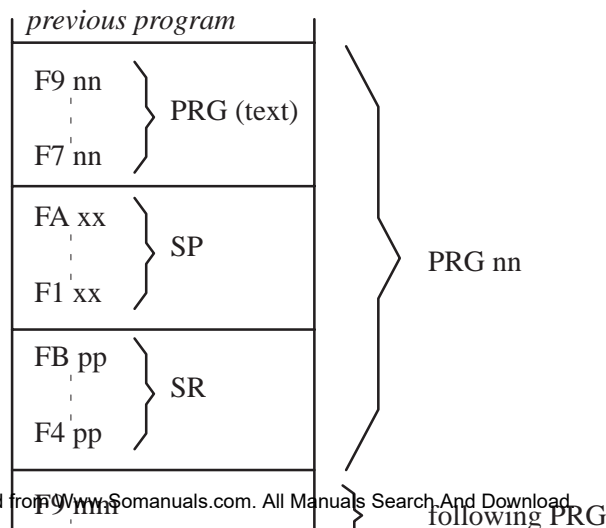
► STEP TRANSITION codes

- 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. Subroutine and program calls**▶ SPECIFIC codes for SP, SR, PLC as an instruction

- 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)

▶ Return label

- 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 – VARIABLE ADDRESSING

### IV – 1. Output – OUT –

Accessible in read and write.

| Number<br>(logical address) | Physical<br>address | Structures / Functions                      |
|-----------------------------|---------------------|---|
| OUT 000<br>↓<br>OUT 255     | 28A0<br>↓<br>299F   | <p style="text-align: center;">not used</p> |

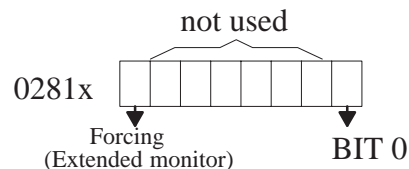
### IV – 2. Input – IN –

Accessible in read.

| Number<br>(logical address) | Physical<br>address | Structures / Functions                      |
|-----------------------------|---------------------|---|
| IN 000<br>↓<br>IN 255       | 29A0<br>↓<br>2A9F   | <p style="text-align: center;">not used</p> |

### IV – 3. User and system bits – BIT –

Each address corresponds to an 8 bit structure in memory.



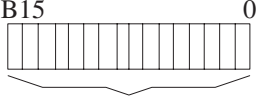
x = bit number in hexadecimal (e.g.: Bit 31, address = 0282F).

Only the low order word is used.

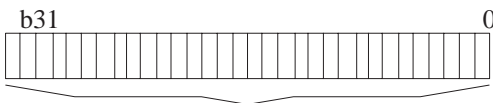
- System bits accessible in Read – No. 0 to 30.
- System bits accessible in Read and Write – No. 31 to 33.
- User bits accessible in Read and Write – No. 34 to 127.

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

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

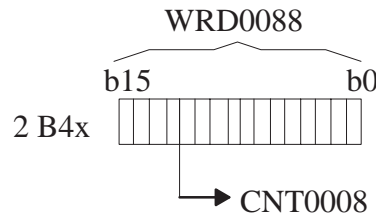
| Number (logical address)  | Physical address  | Structures / Functions   |
|---------------------------|-------------------|--|
| WRD 0000<br>↓<br>WRD 0031 | 2AA0<br>↓<br>2ADF | 32 user Words (read/write) with no predefined functions.<br><br>16 bit structure available |
| WRD 0032<br>↓<br>WRD 0063 | 2AE0<br>↓<br>2B1E | 32 system Words (read only). For the definition of these words, see the Programming Level 2 manual, paragraph I4   |
| WRD 0064<br>↓<br>WRD 0079 | 2B20<br>↓<br>2B3F | 16 user Words (read/write) supporting the PLC timers (TIM 00 to TIM 15).   |
| WRD 0080<br>↓<br>WRD 0095 | 2B40<br>↓<br>2B5F | 16 user Words (read/write) supporting the standard counters (CNT 00 to CNT 15).  |
| WRD 0096<br><br>WRD 4096  | 2B60<br><br>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<br>↓<br>WWRD 063 | 6230<br>↓<br>6327 | 64 user Words (read/write) with no predefined functions.<br><br>32 bit structure available |
| WWRD 064<br>↓<br>WWRD 127 | 6328<br>↓<br>642C | 64 system Words (read only). For the definition of these words, see the Programming Level 2 manual, paragraph I5   |
| WWRD 0116<br>WWRD 0117    | 6400<br>6404      | <i>Specific words</i><br>Values for calculating the automatic anticipated restart.<br>Values for calculating the automatic anticipated restart.<br>See chapter VI – page 28.   |

**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<br>(logical address) | Physical<br>address | Structures / Functions                 |
|-----------------------------|---------------------|--|
| TIM00                       | 2 890               | <p>Only the low order word is used</p> |
| 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.PLC timer**

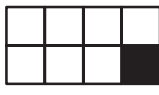











TIM00 to 15 = WRD 0064 to 0079 see chapter IV – 4.

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 declared 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

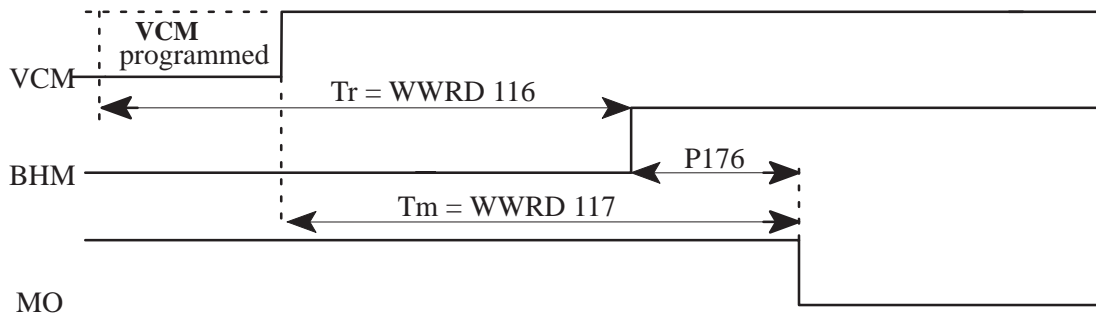
- ▶ Parameter 174 : type of IMM anticipated restart
  - 0 : no anticipated restart
  - 1 : anticipated restart
  - 2 : programmed delay anticipated restart → WWRD 63 programmed in step 0.
- ▶ Parameter 175 : basic value of the auto-adaptative delay and double the minimum value of the programmed delay
- ▶ 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

} in the case of  
restart with  
programmed  
delay

$\frac{\text{parameter 175}}{2}$

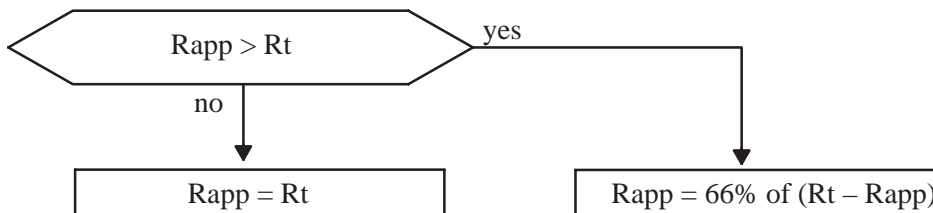


Tr = robot disengaging time in 1/10 s (WWRD 116)

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.

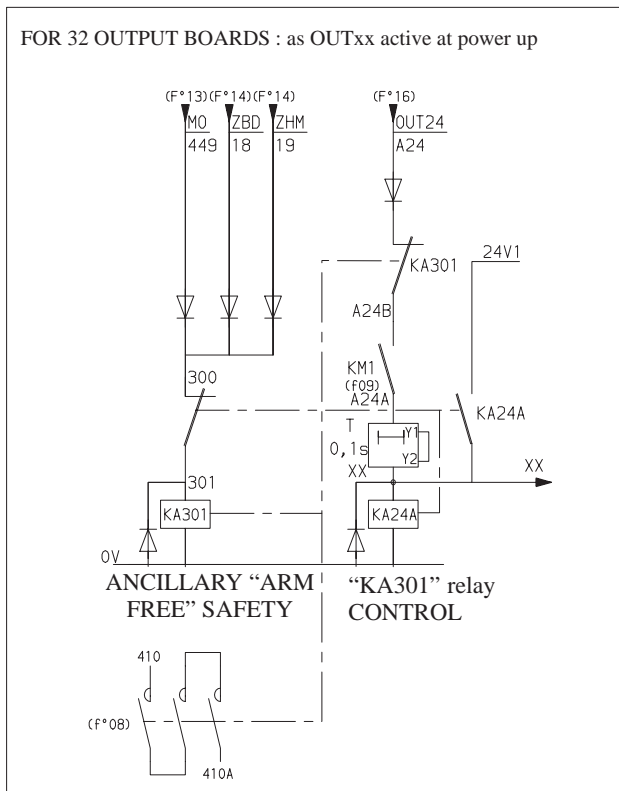
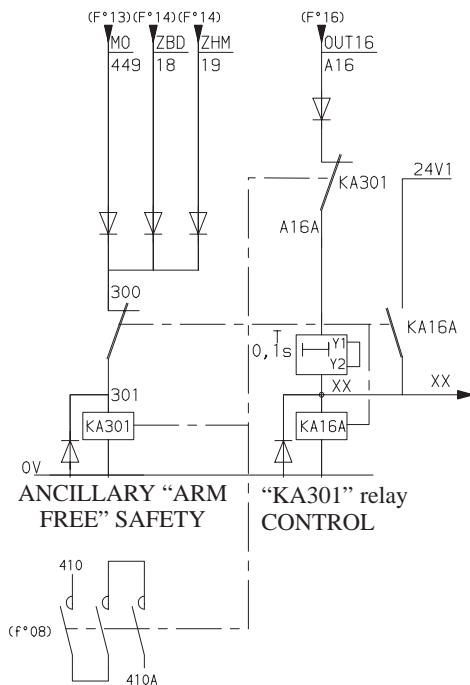
A hard-wired circuit controls the respective positions of the moving mould (“MO” = Mould Open signal) and of the robot (“ZBD” = Arm Free Area / “ZHM” = Outside Mould Area signal).

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 which stops the KA301 relay becoming active (the blocking of KA301 prohibits the IMM cycle).

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





IF IN XX

SET WORD 62 = 200

Until a parameter for the control input for the anticipated restart safety circuit is integrated into the software, this input must be monitored and a fault must be generated using the monitoring PLC.

RELANCE ANTICIPEE NON CONFORME : in French

ANTICIPATED RESTART NOT CONFORM : in English

REARME ANTICIPADO NO CONFORME : in Spanish

VORAUSB. NEUSTART FEHLERHAFT : in German



## Free Manuals Download Website

<http://myh66.com>

<http://usermanuals.us>

<http://www.somanuals.com>

<http://www.4manuals.cc>

<http://www.manual-lib.com>

<http://www.404manual.com>

<http://www.luxmanual.com>

<http://aubethermostatmanual.com>

Golf course search by state

<http://golfingnear.com>

Email search by domain

<http://emailbydomain.com>

Auto manuals search

<http://auto.somanuals.com>

TV manuals search

<http://tv.somanuals.com>