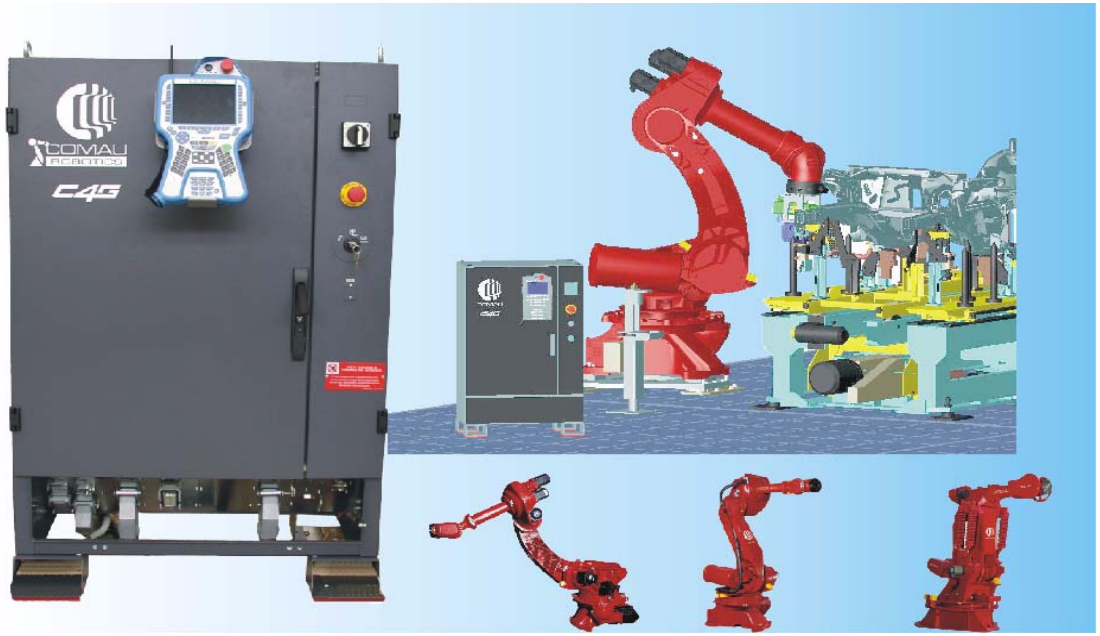
Basic Training Course - C4G Use and Programming

SUMMARY

Basic Training Course - C4G USE AND PROGRAMMING - First Day

1 C4G Robot Control Unit Overview



The C4G Robot Control Unit can manage the entire range of Comau Robots of the “N” series.

The C4G Robot Control Unit

– can handle robots configured with up to 10 interpolated axes, with a continuous overall power maximum 2 kVA, equipped with brushless synchronous motors and high resolution encoder type (Sin/Cos) position transducers.

– is supplied directly from the factory mains with a voltage from 400 Vac -15% to480 Vac +10% with no need for an adapting transformer.

– has a Digital Servo Amplifier Unit to control 600V motors.

– has provisions and options to manage the most common types of Field bus (DeviceNet, Profibus-DP, Interbus-S and EtherNet/IP).

– has a teach pendant for the programmer with a 4096 colour TFT 6.4” graphic display. It has an easy-to-understand user interface, it is lightweight and ergonomically structured with an on-board USB interface.

– has all the most common communication interfaces (USB port, Serial port and Ethernet port). Within certain limits, all these connections can be used for programming, software updating and communication with external devices.

2 Safety Precautions for using the Robot and Control System

Here below are summarized some of the safety precautions: it is strongly recommended to carefully read Chap. General Safety Precautions, included in all standard manuals.

COMAU Robotics & Service shall in no way be held liable for any accidents caused by incorrect or improper use of the Robot and Control System, by tampering with circuits, components or software, or the use of spare parts that are not included in the spare parts list.

The non-observance of the Safety Standards could cause injuries to the operators and damage the Robot and Control System.

– Putting into service is only possible when the Robot and Control System has been correctly and completely installed.

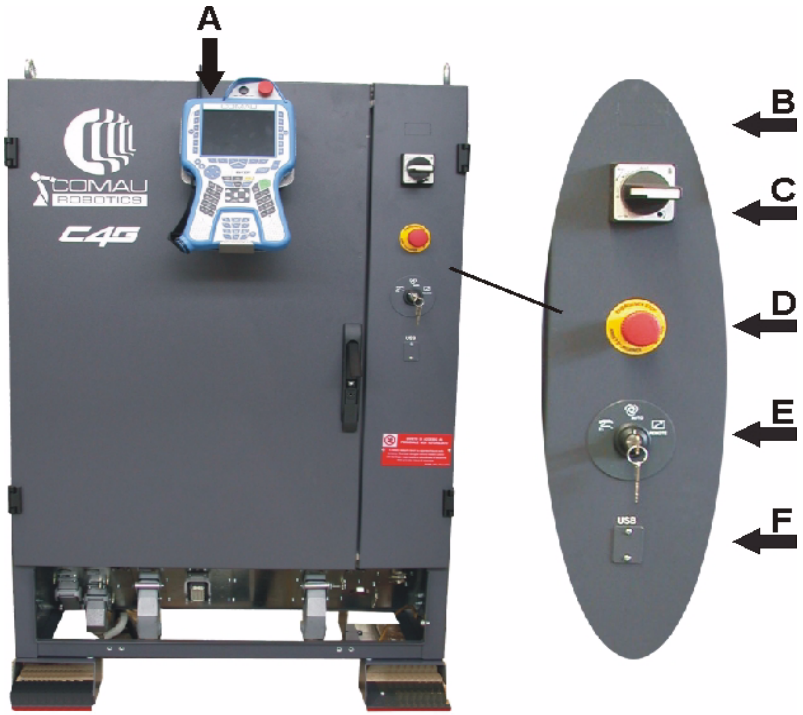
– The robot is only to be programmed by the authorised personnel. Before starting to program, the operator must check the Robot and Control System to make sure that there are no potentially hazardous irregular conditions, and that there is nobody inside the protected area.

– During the programming session, only the operator with the hand-held terminal is allowed inside the Protected Area. Activation of the motors (Drive On) is always to be controlled from a position outside the range of the robot, after checking that there is nobody in the area involved. The Drive On operation is concluded when the relevant machine status indication is shown.

Special attention is to be paid when programming using the hand-held terminal: in this situation, although all the hardware and software safety devices are active, the robot movement depends on the operator.

– The activation of the automatic operation (AUTO and REMOTE states) is only to be executed with the Robot and Control System integrated inside an area with safety barriers properly interlocked, as specified by Safety Standards currently in force in the Country where the installation takes place. Before starting the automatic mode the operator is to check the Robot and Control System and the protected area to make sure there are no potentially hazardous irregular conditions.

3 Operator Panel



In the front you can find:

- (A) Teach pendant TP4i / WiTP
- (C) Main switch
- (D) Emergency stop button
- (E) Modal selector switch
- (F) USB port Provision for USB port
- (B) Provision for hour counter (optional)
- In the lower door you can find:
  - 1 plug x10 - Services and I/O connections on board robot
  - 1 plug x60 - Power (motors and brakes).
  - 1 plug x90 - Field bus (optional) slave, Automatic cycle start and stop command
  - 1 plug x30 - Safety signals for emergency stop devices, line access gates control, etc.
  - 1 plug x124 - Teach Pendant connection
  - 1 plug x93 - Field bus (optional) master

In the back you can find:

- air recirculation system
- cooling fan (lower part)

For further information, see

3.1 Functions of Operator Panel devices

Main switch - Cuts out the power supply voltage to the Control Unit. Wait at least 30 seconds before powering on again.

If the control panel door is opened and closed again, make sure that the control lever and/or the extension shaft of the main switch Q100 are in the same position (ON /OFF) to avoid causing damage to the control lever.

Modal selector switch - Allows to select the Control Unit Command Mode. It has got 3 standard positions + 1 optional position.

- **T1** - The Programming mode (T1) is used to create and verify programs The robot moves, for safety reasons, are run at a lower speed than in automatic mode (maximum robot speed allowed in programming is 250 mm/s on the flange centre). The operator is allowed to operate inside the cell.
- **T2 (optional)** - In Auto T mode (T2 - optional feature) the system runs as in Programming mode (T1), except that the program testing can be executed at working speed.
  The operator is allowed to operate inside the cell.
  Warning! The program ALWAYS starts at low speed, and the user, if wished, CAN increase it to the working speed. If jogkeys are used, the system will AUTOMATICALLY reduce the speed to below the limit of 250 mm/s at flange centre.
  When using with the modal selector switch in position T2 programming control mode the operator is to be very careful, since the risks due to the faster speed of the robot cannot be solved with the normal procedures and precautions.
  Note - When the modal selector switch is set to T2, the system generates a latched alarm that prevents the entry to the actual T2 status, even if allowing to switch the drives on. No movements are allowed (nor the manual one) until the latched alarm is not acknowledged.
- **AUTO** - Local automatic mode (AUTO) is used to execute production programs; as they contain instructions for the robot movement, to be able to start it is necessary to press the START key on the Teach Pendant. The modal selector switch must be set on AUTO.
  The operator is NOT allowed to operate inside the cell.
- **REMOTE** - The Automatic remote mode (REMOTE) is the same as Automatic local mode (AUTO), but the commands (for example the start) are sent from a remote device (for example a PLC).
  The operator is NOT allowed to operate inside the cell.

The selector switch key can be removed in all positions.

Emergency stop pushbutton - To stop the machine immediately. When pressed it cuts out the power to the robot motors and activates the brakes. It has forced opening contacts with mechanical latching. It has to be pulled to reset. According to the selection on the JP15 jumpers of the RSM board, the AUTO or REMOTE mode stop takes place: – in controlled mode (class 1 as per standard EN 60204-1) – immediately (class 0 as per standard EN 60204-1). If the controlled stop function class 1 (EN 60204-1) is active, the power cut-out (opening of the power contactor) may take place with a delay that ranges from a minimum of 1 second to a maximum of 2 seconds. In T2 or T1 mode, the stop is always immediate (class 0 as per standard EN 60204-1).	
USB port (optional feature) The USB port is used to connect an external storage device, to load and save the programs from Control Unit to PC and vice-versa. The Teach Pendant has specific controls to activate the file transfer. The device is automatically acknowledged a few seconds after insertion and can be extracted without any specific operations. To ensure correct file transfer it is not to be extracted during the transfer.	
Hour meter (optional) - Where present, it displays the number of operating hours (in DRIVE ON) of the Controller and the Robot.	

4 System states

The current system status is displayed on the first status line of the Teach Pendant (or in the Terminal window of tool WINC4G on PC).

The system status depends mainly on:

– the modal selector switch on the Robot Controller Cabinet (RCC),

– the DRIVE ON, DRIVE OFF and HOLD keys on the Teach Pendant,

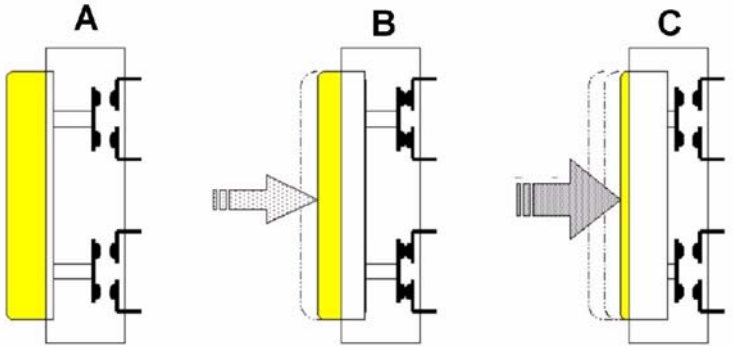
– system alarm.

– enabling device - Transition from one state of the system to another is also influenced by the enable device on the Teach Pendant. On the rear of the Teach Pendant there are two pushbuttons for the Enabling Device. See the low left figures.
Each of these is a three-position safety device that is to be kept pressed in the intermediate position, to allow movement in automatic or in manual mode, when the system is in Programming mode. When this pushbutton is pressed the motors are activated automatically (DRIVE ON).

Enabling Device operating mode

The type of operation for each of them is as follows:
– A - released - Drive OFF
– B - intermediate pressure - Drive ON
– C - fully pressed - Drive OFF (anti-panic)
– pressing both these pushbuttons at the same time is interpreted as an error by the system, therefore only one at a time is to be used.

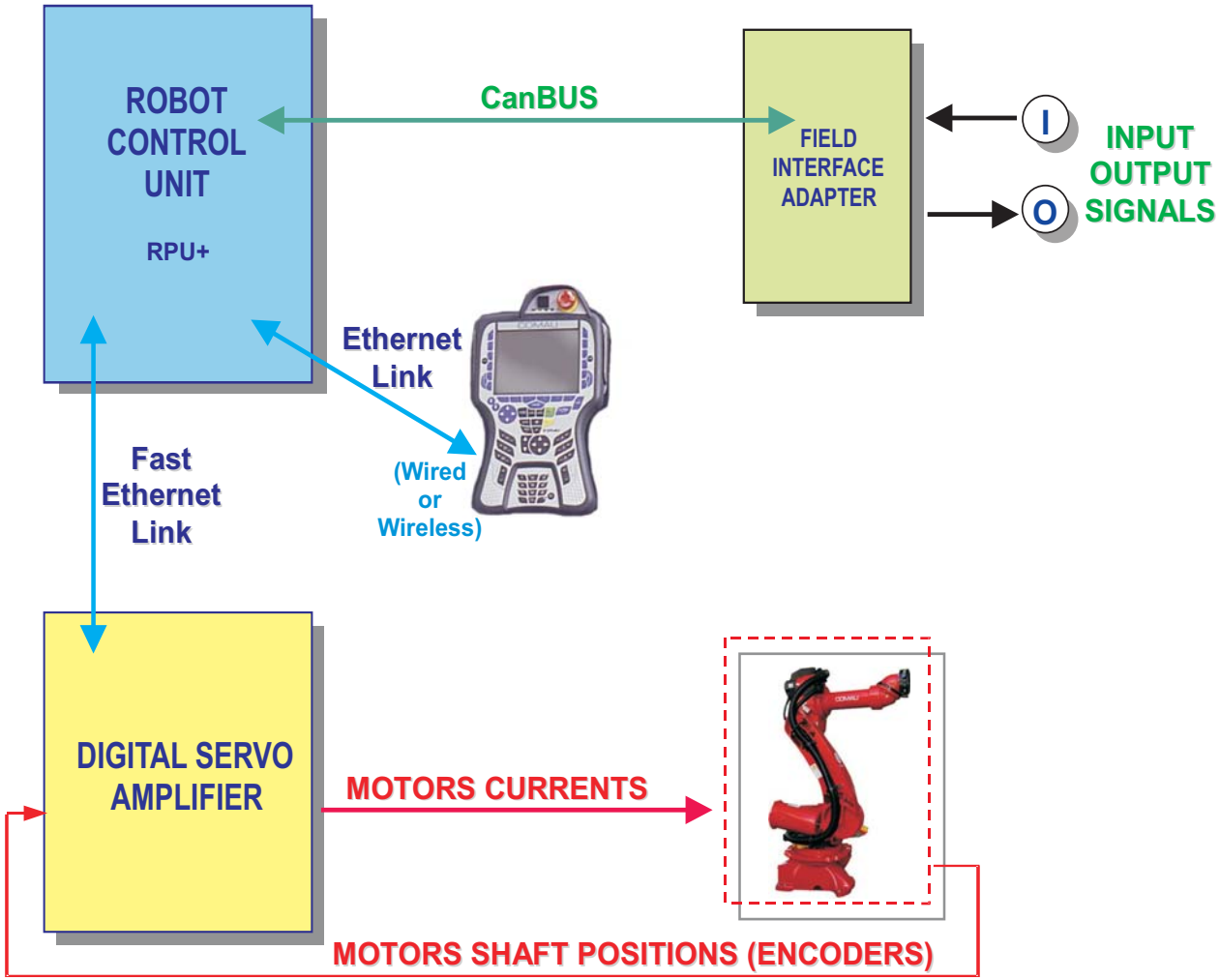
The right pushbutton and the left pushbutton operate in exactly the same way. The purpose is to have an Enabling Device pushbutton for both right-hand and left-hand operators.



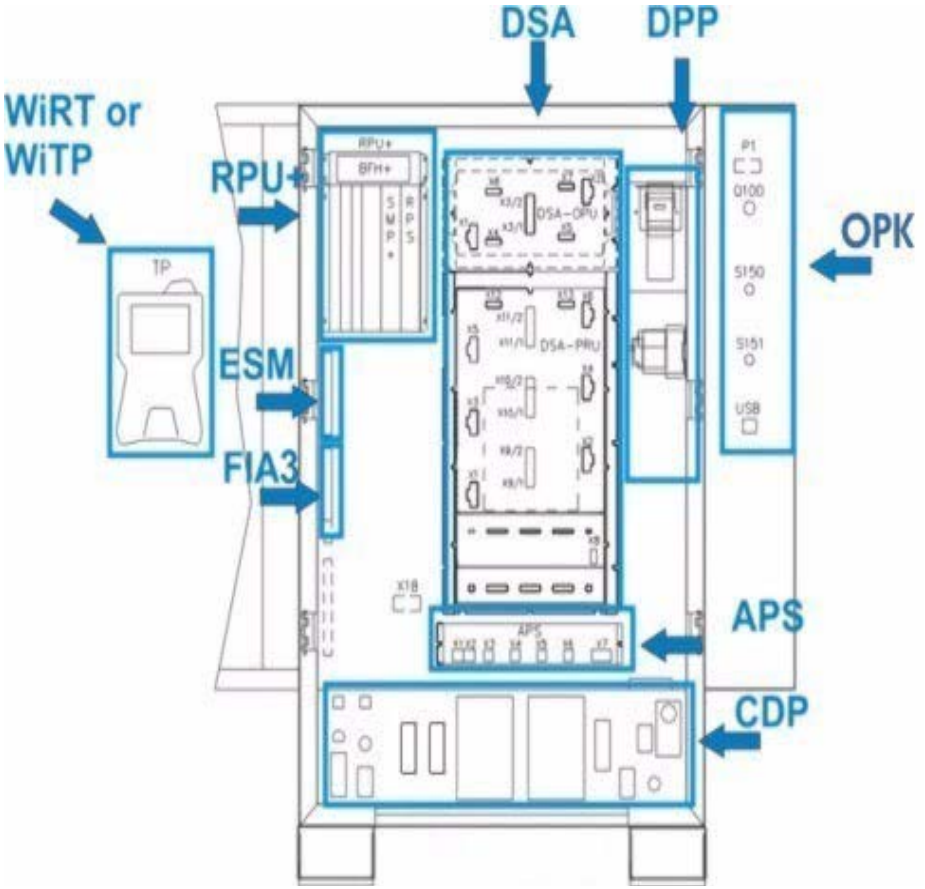
The Control Unit may be in one of these conditions:

- **HOLD**
  HOLD status: the robot is gradually decelerated until the stopping point is reached; movement is suspended and also the execution of the movement program (holdable). When there are all the necessary conditions to exit from the HOLD status, the system returns to the previous state (programming or automatic), but to continue to execute the movement program it is necessary to press START.
  If the HOLD status has been caused by pressing the DRIVE OFF key on the Teach Pendant (either Enabling Device released or R5 softkey pressed meaning DRIVE OFF), the DRIVE OFF and HOLD keys must be pressed again to exit from HOLD status, and then re-power the drives (either intermediate pressure of the Enabling Device or press R5 softkey meaning DRIVE ON).
- **AUTO**
  AUTO status: this is usually used to execute production programs that control the robot movements (modal selector switch positioned on AUTO or REMOTE or T2). In AUTO status, to start programs ready for execution, press the START key on the Teach Pendant or activate the START input from remote device.
  When in HOLD status, to switch to AUTO bring the selector switch back to the required position and, if needed, power-on the drives (DRIVE ON) and press the HOLD button.
- **PROGR**
  In this state the robot can be moved manually, using the jog keys on the Teach Pendant. It is also possible to run programs from IDE environment (see IDE Page in Use of C4G Control Unit manual) to check that they are correct and if necessary make changes. Movements are at slow speed.
  To be able to move the robot arm, both the START button and the Enabling Device must be kept pressed. In this status, the modal selector switch is set to T1.
- **AUTO-T (OPTIONAL)**
  In this status the movements can be run, at full speed, from the Teach Pendant, requiring that the START key, together with the enabling device, is kept pressed by the operator to execute the move.
  This status is active when: modal selector switch is set to (T2) and latched alarm confirmed.
- **ALARM**
  ALARM status: this status is entered when there is a system alarm. According to how serious the error is, the system takes different actions, such as suspending the program execution, deactivation of the drives, etc. A situation may occur where the alarm cannot be reset, therefore the drives cannot be switched on.

5 Hardware Structure Overview



Location of modules in the Control Unit



The C4G Control Unit is composed of the following modules.

- DPP (Distribution Power Panel)
- APS (Auxiliary Power Supply)
- RPU+ (Robot Processing Unit Plus)
- DSA (Digital Servo Amplifier)
- FIA3 (Field Interface Adapter)
- ESM (Electronic Safety Module)
- Operator interface: the devices that the operator uses to interact with the Control Unit; including
  - Operator Panel
  - Teach pendant
  - WinC4G Program
- CDP (Cabinet Distribution Panel)
- Cooling system

For further information, please refer to the Connection architecture and principle Block Diagram

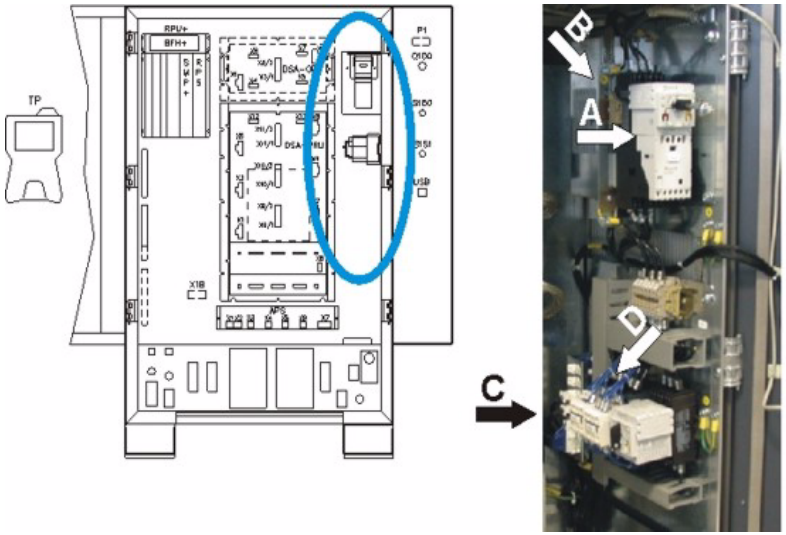
5.1 DPP (Distribution Power Panel)

The DPP is the group of panels that supports the electromechanical power components of the system.
It distributes the electrical power to the auxiliary circuits power unit APS and to the DSA digital drive.

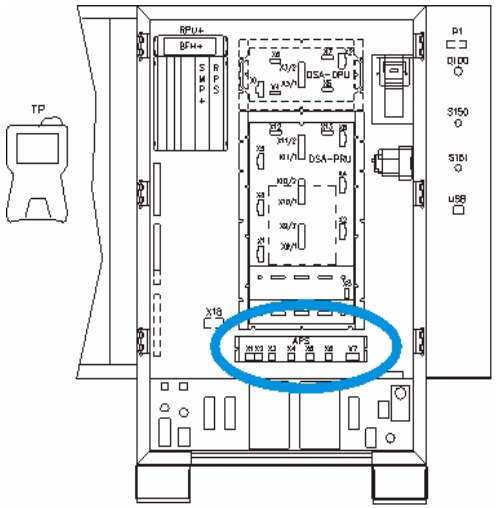
The DPP basic version consists of:

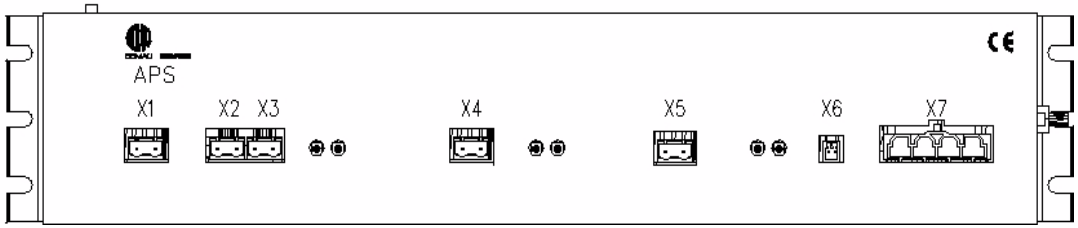
– main switch (A)
– network filter (B)
– electric power circuits (C) that manage the network power supply to the APS and DSA devices.
– power contactors (D).

The DPP is installed inside the electric cabinet, on panels located on right side.



5.2 APS (Auxiliary Power Supply)





The APS is the main power unit inside the Control Unit. It is switching type and supplied with a three-phase voltage from the DPP.

It is the Auxiliary Circuits power Unit: it supplies the needed voltages to RPU, DSA submodules and interface modules available in the Controller. It supplies 24 Vdc for all the auxiliary circuits (RPU, DSA, FIA), except the brakes power supply which is 25 Vdc.

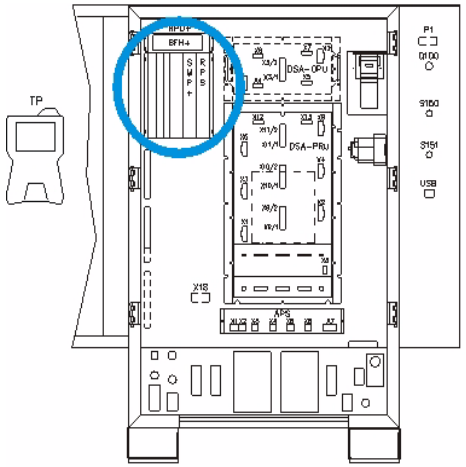
The outputs can be monitored by the green and red LEDs located near the output connectors.

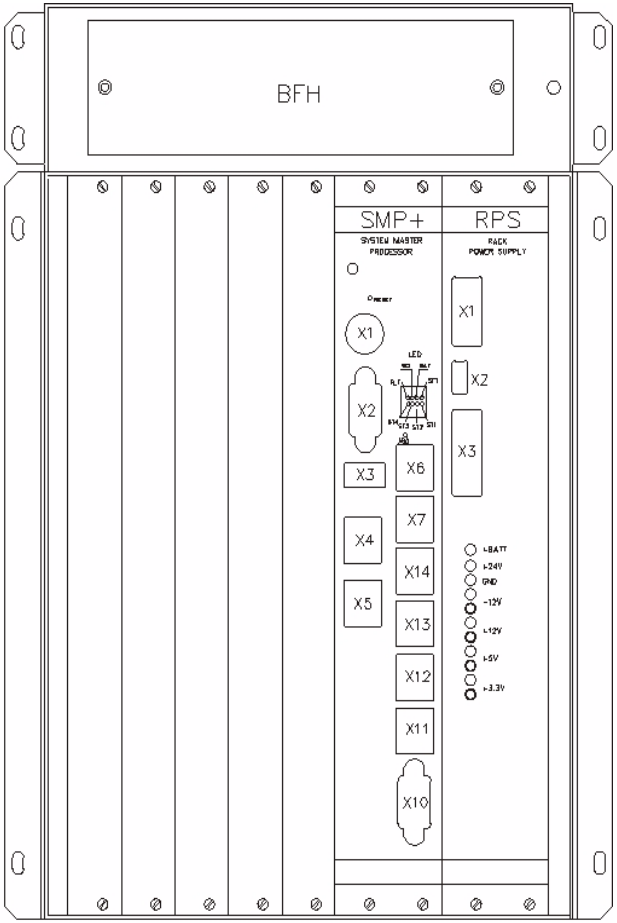
The power failure signal coming from connector X6 of the power unit signals there is no power supply coming in from the mains.

The power unit is protected against overloads, short circuits over/under voltage independently on the separate outputs.

If the overload/ short cicuit or output voltage out of range triggers, the related green LED switches off and the red LED switches on.
The protection resets itself when the overload or short circuit is removed. The indication with the red LED remains even with the connectors disconnected in the case of output voltage out of range.

5.3 RPU+ (Robot Processing Unit Plus)





The RPU+ is the main control module for the entire system. It handles the information traffic in the Control Unit, including networks, programs and dialogue to the drive and the safety boards.

It is the main controller unit of the whole system, consisting of **RPS** and **SMP+**.

Compared to the previous versione of the Controller Unit, with two CPUs (SMP and MCP), in the current versione there is just ONE CPU (SMP+) which performs all the functions.

The RPU+ co-ordinates and supervises the Robot control activities, among which:

– robot and application program execution
– I/O and field bus management
– motion control.

The system software, programs written by the user, positions recorded when programming the robot and the motion management software are processed in a single CPU SMP+.

The archiving of the software and the data takes place on a removable Compact Flash, installed in the CPU SMP+. This solution speeds up the system reset if boards are replaced.

The RPU+ communicates with the other system modules through these serial lines:

– Ethernet
– CAN Bus
– field bus
– USB
– serial line RS422
– Sync - DSA.

The boards in the RPU are developed in conformity with the Compact PCI bus standard, are connected to each other by back-plane and interfaced with Comau-owned software.

Commercial boards not specifically indicated by Comau cannot be installed unless they have been previously integrated with the software owned by Comau. This implementation can be obtained upon request.

The RPU consists of a rack of 6 compartments with 5 slots for the Compact PCI boards and a slot occupied by the power unit and a support for the fans and the battery.

The rack, in the standard version, consists of:

– RCK (Rack Boards with back-plane): wrapping, including back-plane, to contain all the boards

– RPS (Rack Power Supply): supplies the functioning voltages required for the boards in the rack. The RPS is supplied by the 24 Vdc coming from the APS. It is item (A) in the figure on the right.

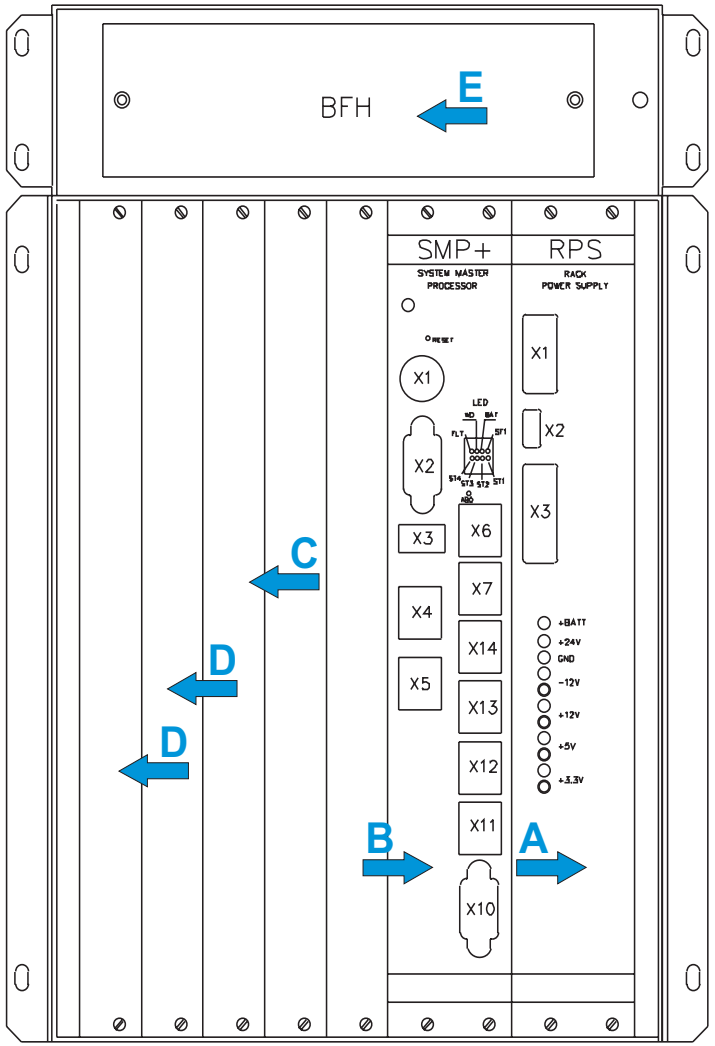
A - RPS power unit

B - CPU SMP+

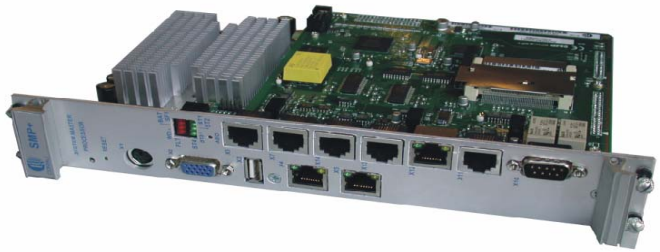
C - Slot 4, available for the user

D - Slots 5 and 6, available for the user, usually used to install field bus and application boards (SWIM_4G)

E - Fans and battery compartment



– SMP+ (System Master Processor): is the CPU of the system. Includes the operating system, I/O management, user interface management, Ethernet, CAN Bus and RS422 serial communication, software for the PLC section and the application programs, trajectory and the Ethernet communication lines management and synchronism with DSA drive.
The software is on a 64 Mbyte Compact flash (CF64).The Compact Flash is supplied with the software already loaded and cannot be substituted with a standard blank Compact Flash
The SMP+ board also monitors the state of the rack cooling fans and warns of any malfunctioning with diagnostic messages. The board has internal PMC housings for future expansion.



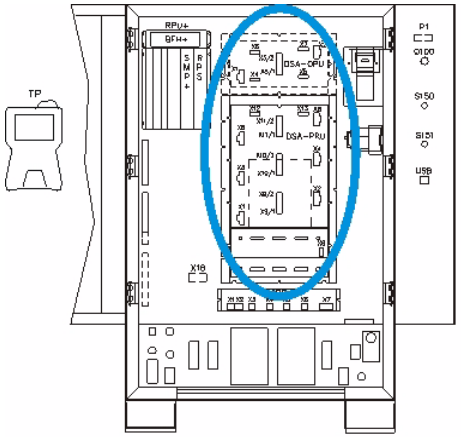
– Free slots: 3 free slots for the insertion of optional boards to manage outfittings or field bus. These boards must be Compact PCI compatible.
In slot 4 boards can be installed that use only connector J1 or owner boards. (Item (C) in the figure in previous page))
Any type of board can be installed in slots 5 and 6. It must be considered that J1 is the only connector connected to the bus. (Item (D) in the figure in previous page)

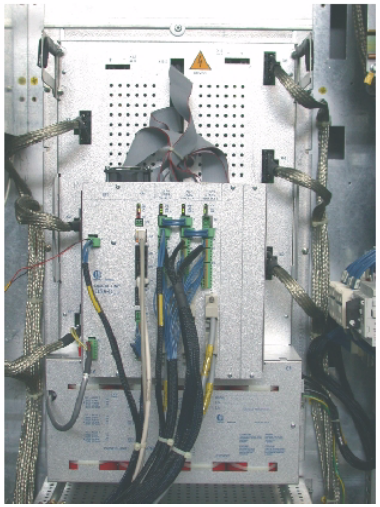
The fans and battery support consists of (item (E) in the figure in previous page):

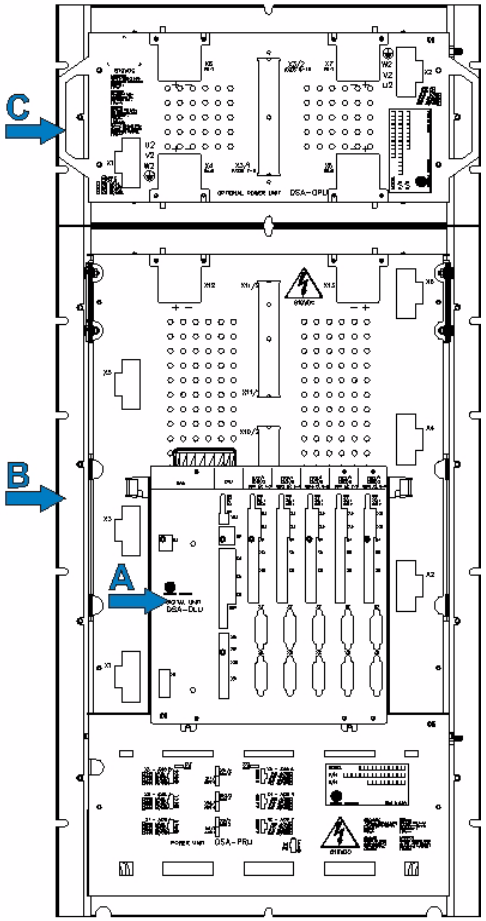
– BFH+ (Battery Fan Holder): a wrapping with fans and a battery housing. The fans, installed on the top of the rack, ensure the necessary localised cooling of the boards in the RPU
– Battery: a pack of Nickel Cadmium batteries connected directly to the power unit (RPS). The buffer battery is used to ensure the storage of the robot position and the state of programs if there is an unexpected power failure. The data storage activity lasts less than 30 seconds and is executed to ensure that the robot starts functioning again from the point where it stopped. The battery is automatically recharged while the system is in use.

The RPU is installed on the front panel of the electric cabinet, on the left.

5.4 DSA (Digital Servo Amplifier)







The DSA is a digital drive that can control up to 10 axes. It is a system with high potentiality, able to manage the axes locally closing the position, speed and current loop. It is supplied by the mains, integrates the recovery resistance, soft start and dynamic braking. It controls and supplies power to the robot motors.

It manages and co-ordinates the robot movements using the encoders for movement feedback. As an option there are control boards for use of motors with resolver.
The dialogue between DSA and RPU is via Ethernet consisting of the traditional connection integrated with a synchronism signal to ensure the dialogue synchronizing between the two systems.

The standard configuration consists of 6 axes with power modules having different current thresholds according to the type of robot, that can be expanded up to 10 axes with additional options.

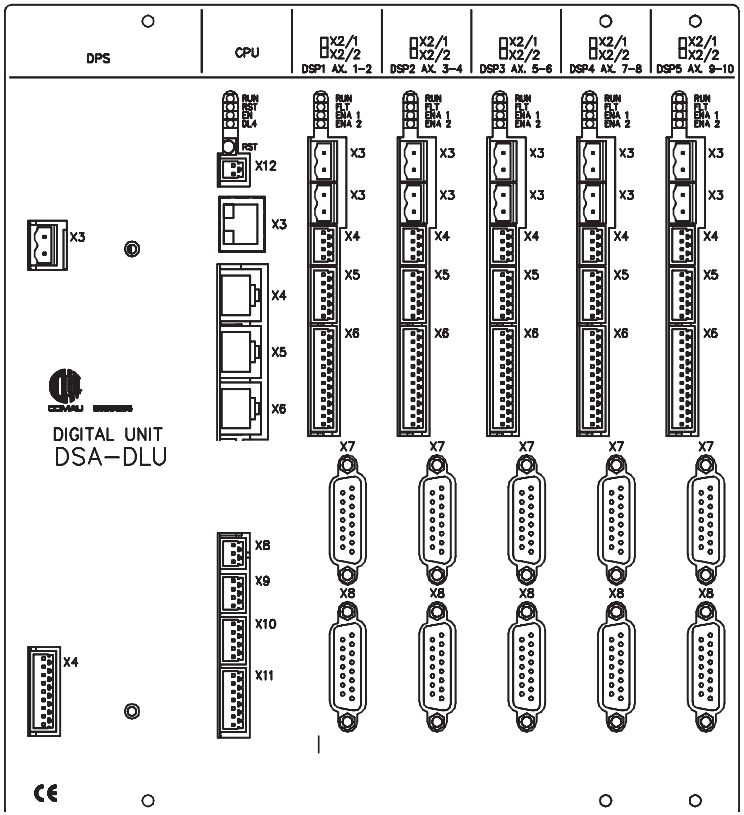
The DSA consists of:

– DLU (Digital Logic Unit) logic unit that controls position, current speed / torque (A)
– PRU (Power Unit) power stage for 6 axes (on standard version) (B)

The optional expansion for 2 additional axes is OPU (Optional Power Unit) (C)

5.4.1 DLU (Digital Logic Unit)

DSA-DSP connections
X3 - 24 Vdc Input
X4 - DSA-PRU connection



The picture shows the rack complete with 5 DSA-DSP boards for axis control

DSA-CPU connections

X12 - *1

X3 - Ethernet port connected to SMP+

X4 - Serial port for diagnostics

X5 - Ethernet synchronism to other DLUs

X6 - Ethernet synchronism connected to SMP+

X8 - port for diagnostics

X9 - Safety contacts

X10 - digital Outputs

X11 - digital Inputs

DSA-DSP connections

X3 - 24V input to supply brakes

X4 - Brakes command

X5 - Axes overtravel limit switch

X6 - HSI / analog digital inputs

X7 - Encoder A (for first axis)

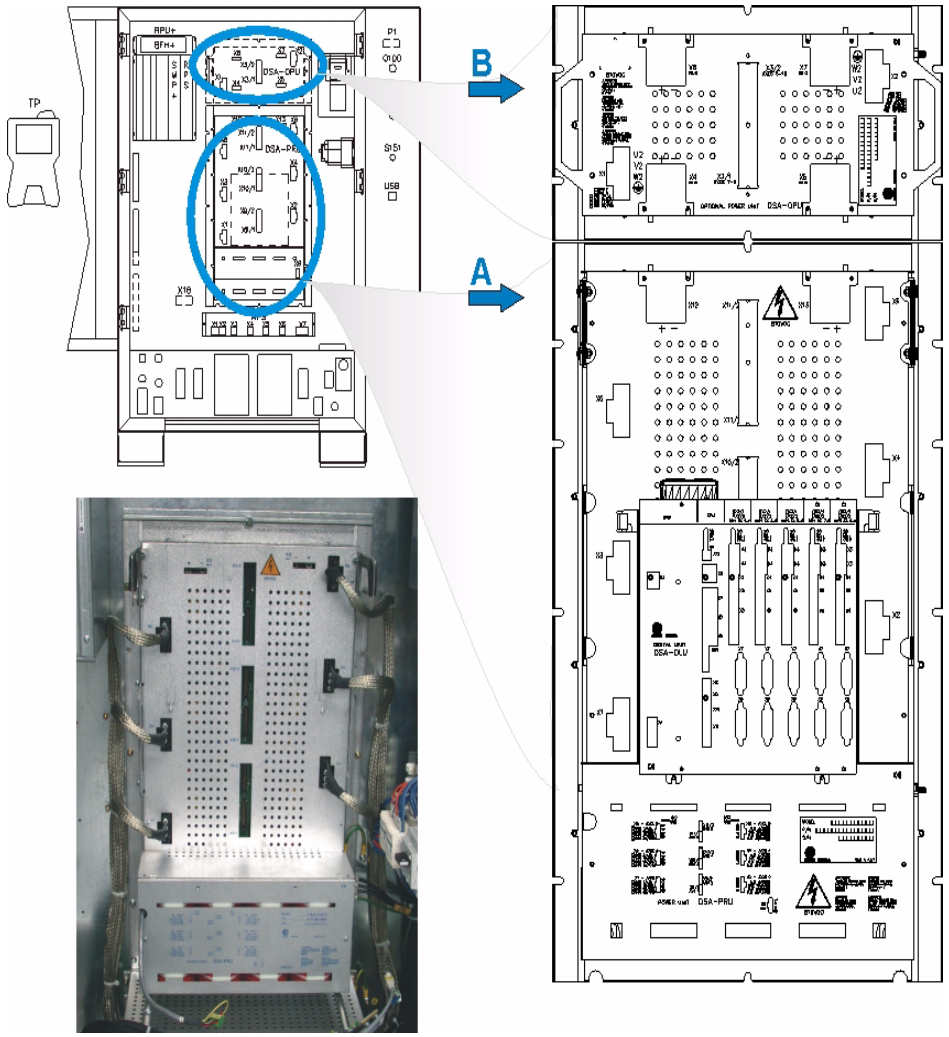
X8 - Encoder B (for second axis)

The logic control unit is a module that consists of:

- DSA-DCH (Digital CHassis): rack, with back-plane, to house the boards.
- DSA-DPS (Digital Power Supply)
- DSA-CPU: CPU to control the drive.The board has an IP-Address for the network addressing of the DLU. The address can be changed for any expansions.
- Axis board DSA-DSP-2AX. Each board controls 2 axes. In the standard version there are 3 boards to manage 6 axes
- 2 empty slots for the expansion of another 4 axes.

*1 Connector or function reserved for Comau operators
*2 Digital inputs/outputs on service connectors on board Robot

5.4.2 PRU (Power Unit) and OPU (Optional Power Unit)



The power stage is divided into 2 sections:

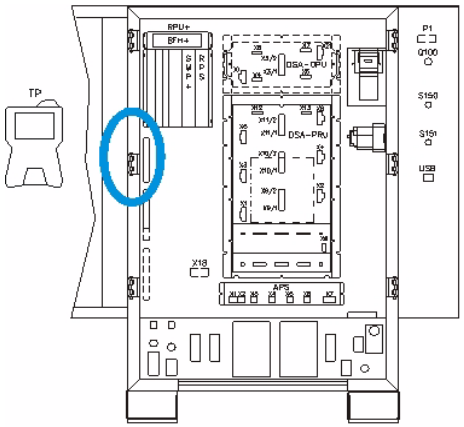
– main section PRU, containing 6 power modules for axis control (A), power unit MPS, recovery resistor and bank of capacitors

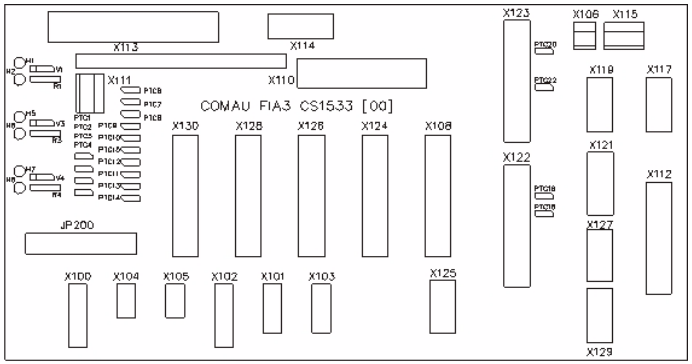
– expansion section OPU: housing for the installation of the 7th / 8th axis (B).

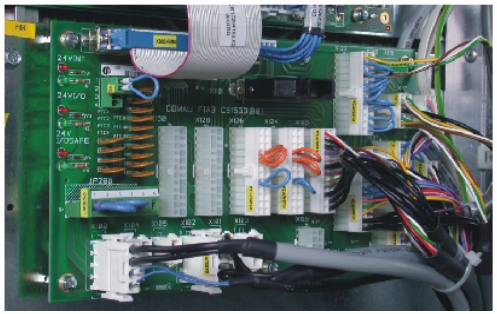
The installation of the 9th / 10th axis is possible only if extra steelwork structure is added on the roof of the Control Unit.

The connection to the robot is executed for each axis on Control Unit base, through connector X60.

5.5 FIA3 (Field Interface Adapter)







The FIA3 is a board that collects and distributes all the signals in the Control Unit (control status signals, safety signals, signals from and for the application, 24 Vdc power supply).

It also contains the distribution protections of the 24Vdc power supplies.

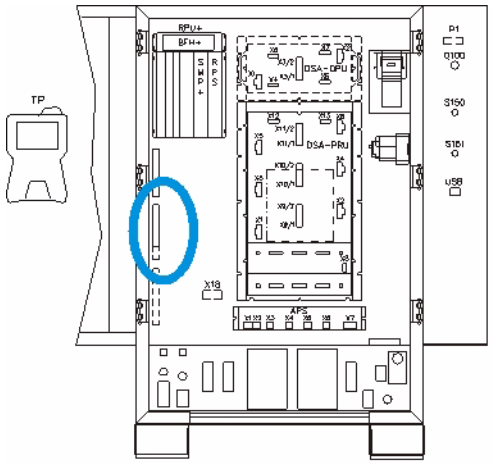
The main connections of the Control Unit auxiliary circuit are by means of the printed circuit on the FIA3 board.

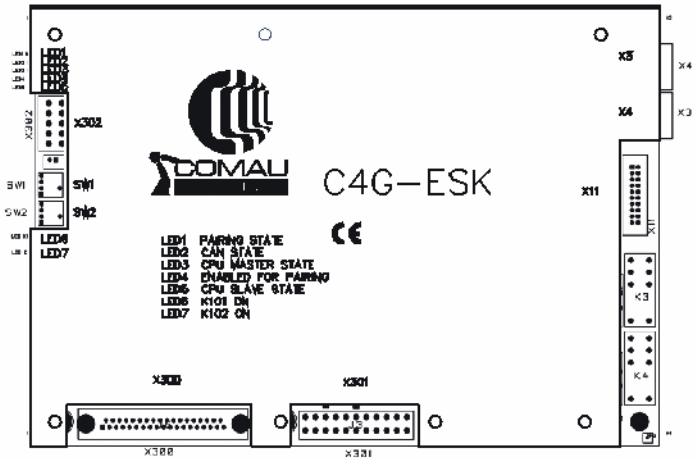
Connectors are available on the board to expand the Control Unit functions.

The FIA3 board has all the protections installed against overload and short circuit of the various 24 Vdc power supply circuits , except for the 24V for the field bus. The protection is by means of self-resetting fuses (PTC) that do not have to be changed after they have intervened.

The FIA3 board is installed inside the electric cabinet, on the left-hand panel. See fig. on the left.

5.6 ESM (Electronic Safety Module)





The ESM board is installed inside the electric cabinet, on the left-hand panel. See fig. on the top left.

The ESM is sold as a spare part with the code ESK, that also includes the supporting metal base.

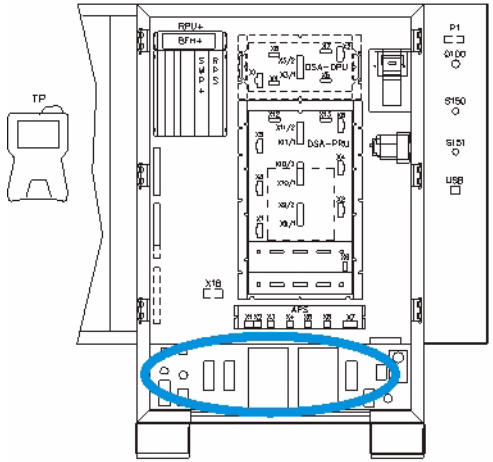
The ESM board manages the C4G input and output safety signals: emergency stop (internal mushroom head pushbuttons and contacts coming from the line), safety gates, general stop, operating mode selection, enabling device and power contactors.

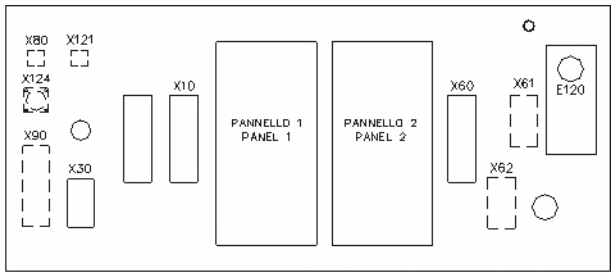
The safety signals are managed directly by the board through two self-controlled processors. Dialogue with CPU SMP+ is guaranteed by a specific protocol transmitted on CAN Bus network.

The input/output signals coming from and directed to the cabinet and the line are connected by opto-isolators and relays.

The ESM board is in conformity with category 3 as per standard EN ISO 13849-1. The same category is kept on the Control Unit internal circuits. The emergency stop category (as per standard EN 60204-1) is configured on 0 in programming mode and 1 in automatic mode. The stop circuit times can be changed through the rotary switches on the board.

5.7 CDP (Cabinet Distribution Panel)





The distribution panel contains all the connectors and connecting cables from and for the Control Unit.

The CDP is installed at the base of the cabinet, protected by a removable access door. See fig. on the left.

The panel contains the following connectors:

- X10 (signals) and X60 (power) for connection to robot.
- X30 for line connection.
- X90 to pass the cables outwards. The connector is a 4-hole cable fairlead type.
- X124 for connection to the Teach pendant (in the version via cable only).

There may be other connectors present, according to the options installed:

- from X31 to X34 for connection to applications.
- X93 for field bus input (output to connector).

There are also removable caps for the connection of additional connectors and/or cables.

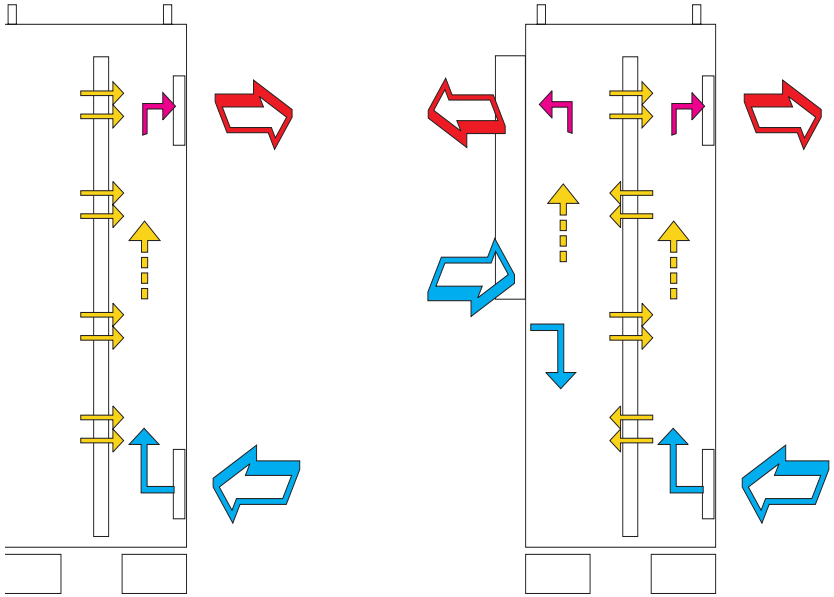
5.8 Cooling system

The Control Unit has a standard cooling system installed that consists of a fan located in the rear of the electric cabinet base.

The position of the internal components, the fan and the air transit channel in the cabinet allow appropriate cooling of the drives and all the electric cabinet in the basic version. This condition is guaranteed when the cabinet is installed in an environment with a temperature from 5°C to 45 °C (41 °F to 113 °F).

In configurations with a power installed that is more than 9 kVA or Robot and special applicationse that require a higher heat dissipation it is necessary to install the Fan C4G-SFK (optional). This fan is already installed in some C4G models.

The fans are powered with 24 Vdc generated by the APS.



The position of the fans and the air passage duct have class IP 2x protection. The rest of the cabinet has class IP 54 protection. IP 44 fans (optional) are available. If required by the environment where the system is installed, a Cooling by Air exchanger or air conditioner (optionals). Models C4G RCCxP of the Control Unit are fitted with Air exchanger as standard.

5.9 Teach pendant

The teach pendant is the main operator interface of the Control Unit. It is used to handle, program and learn the positions and integrates control, display and edit functions. It includes safety devices and emergency stop button.

It consists of:
- colour display
- keyboard with function keys, robot movement keys
- motors on/off pushbuttons, enabling and stop pushbuttons
- USB port for Disk On Key (optional) backup unit.

The following solutions are possible:
- **C4G-TP4i teach pendant** - The C4G-TP4i teach pendant has a cable for connection to the Control Unit. With a new Control Unit always pair the Complete station, available with cables of different lengths. It is however possible to later change the teach pendant with a version having a different length cable.

The Teach Pendant solution with cable requires the continual presence of the teach pendant with the Control Unit, otherwise it is not possible to start-up the system.

- **C4G-WiTP wired teach pendant** - The C4G-WiTP wired teach pendant has a cable for connection to the Control Unit . With a new Control Unit always pair the Complete station, available with cables of different lengths. It is however possible to later change the teach pendant with a version having a different length cable. The Teach Pendant solution with cable requires the continual presence of the teach pendant with the Control Unit, otherwise it is not possible to start-up the system.

WiTP wireless

For safety reasons, the C4G-WiTP teach pendant has severe controls on the pairing and requires special behaviour when in use.

– **C4G-WiTP wireless teach pendant** - The C4G-WiTP teach pendant (also called C4G-WiTP-AP) connects to the Control Unit in Wireless mode, with a Docking station installed on the cabinet door. Without a connection cable the teach pendant is easier to handle, with freedom of movement. The range of action of the Teach Pendant is guaranteed up to 40 m / 130 ft.
The C4G-WiTP teach pendant is supplied by battery; it has autonomy for 5 hours and requires 2 hours for recharging (typical times).
For use, the C4G-WiTP teach pendant has to be paired to the Control Unit (Pairing operation); before being used on other Control Units it has to be first unpaired (Unpairing operation) then paired again with the new Control Unit.

The Pairing and Unpairing operations can be executed without stopping the Robot: all the functions, stop pushbutton and Enabling Device are automatically activated/deactivated.
When it is neessary to use the Teach Pendant functions, it obviously has to be paired.
The wireless teach pendant solution does not require the continual presence of the teach pendant with the Control Unit: the C4G-WiTP teach pendant can be one for several cabinets.

With the first new Control Unit always pair the Complete station. On subsequent Control Units it can be decided to pair the Docking station and share the same wireless Teach Pendant or pair Complete station / Docking Station / WiTP Teach Pendants in a quantity that guarantees sufficient flexibility of use and saving on teach pendants.

Note that the listed above solutions for the Teach Pendant are NOT interchangeable!

6 Teach Pendant Main functionalities

The following sections give information about how to use the Teach Pendant, both for TP4i model (wired connected to the C4G Control Unit) and WiTP model (either wired or wireless connected).

6.1 Keys

The Teach Pendant keyboard is basically arranged as follows:

Blue keys - They are divided into:

- Function keys
  The function keys are used to activate the associated softkeys belonging to the different Menus.
  - Left Menu keys (L1..L6)
  - Right Menu keys (R1..R6)
  - Bottom Menu keys (F1..F6)
- General use keys
  - SHIFT - still in combination with other keys; the use changes according to the environment and the function of the key it is associated to
  - MORE - when the softkeys of the Left Menu and/or Right Menu are more than 6, the associated MORE key allows the scrolling of them all
- Navigation keys
  - Page Up and Page Down keys - They move the cursor respectively to the beginning and to the end of the screen page depending on the context in which they are used.
  - Cursor keys (up arrow, down arrow, right arrow, left arrow): to move inside the screen page fields.
  - ESC key: returns back, cancelling the current action
  - ENTER key: confirms the current action
  - SHIFT keys (left and right): can be used in combination with other navigation keys.

TP4i

WiTP

Black keys - These keys are for robot motion.

– **Motion keys**, for both the TP models, are:

- BACK key - Causes the movement backward, to the start position of the current movement, during step-by-step checking of a program.
- COORD key - selects the reference system:
  JOINT - joint mode. The keys are associated to each of the axes of the selected arm; any auxiliary axes, if present, follow those of the arm. Pressing one of such keys determines the movement of the corresponding axis in positive or negative direction, according to the direction indicated by the plate on the arm.
  BASE - linear movement mode according to the world reference frame x, y, z (the workshop reference frame). The first keys allow linear movement in the direction of the three axes of the world reference system, the next three keys allow the tool rotation around the same axes keeping the TCP position unchanged.Note that the world frame is not directly defined by any system variable; in fact, it is the robot base that is represented as to the world, through the $BASE variable
  TOOL - linear movement mode according to the tool reference frame x, y, z (or TCP frame).The first three keys allow linear movement in the direction of the three axes of the tool reference frame (defined by $TOOL predefined variable); the next three keys allow tool rotation around the same axes keeping the TCP position unchanged (tool work point).

TP4i

WiTP

UFRAME - linear movement mode according to the user reference frame x, y, z (for example the set of three that describes the workpiece being machined).The first three keys allow linear movements in the direction of the three axes of the user reference frame (defined by $UFRAME predefined variable); the next three keys allow tool rotations around the same axes keeping the TCP position unchanged
WR-BASE, WR-TOOL, WR-UFRAME: if pressed together with the SHIFT key, the COORD key allows passing between Cartesian movement modes (BASE, TOOL, USER) and Wrist JNT (X,Y,Z and joints of axes 4,5,6). For further information see the chapter ROBOT MOTION IN PROGRAMMING MODE of the Motion Programming Manual.

- Keys +% and -% to change the OVERRIDE. Combined with the SHIFT key, the following values can be obtained:
  SHIFT -% --> 25%
  SHIFT +% --> 100%

- JOG keys - Such keys can be used to move the robot axes. AUX keys can be configured for axes 7, 8, 9, 10, two at a time

– JPAD - The JPAD group allows jogging referred to the USER position. Furthermore:

  - The two cursor keys on the left refer to moves along axis z: upward and downward movements

  - the four keys on the right are respectively for movements along x axis (upward arrow and downward arrow) and along y axis (left arrow and right arrow). The movements along x axis are for moving away, and for approaching the user; the movements along y axis are for moving left and right, always referred to the user

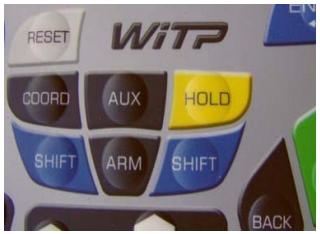
The position of the user can be configured on sub-page Advanced of the Motion Page.

It is to be remembered that any move executed using JPAD is always according to user reference ($UFRAME), except for movement of Gantry with 3 linear axes, which is always a joint move.

TP4i

WiTP

– **Motion keys**, specific for each TP model:



WiTP

- (***WiTP***) AUX key - pressing this key increases (in circular mode) the AUX-A index, the key that moves auxiliary axis A. When pressed together with the SHIFT key (SHIFT+AUX) it increases (in circular mode) index AUX-B, the key that moves auxiliary axis B. The values of AUX-A and AUX-B are displayed on the Motion page, sub-page Coop. The same information is displayed in the Status bar, in the Arm field (fourth field).
- (***WiTP***) ARM key - corresponds to the Arm softkey of the TP4i Right Menu (see ARM (R1)). In Multiarm systems, it is used to manage the index of the main Arm and the synchronised Arm, displaying in the Status bar, in exactly the same way as by entering in the Motion Page, sub-page Basic and modifying the Arm field.The functioning is as follows:

  In DRIVE-OFF - pressing ARM increases in circular mode the index of the main Arm, but never altering the Arm quantity in the Status bar (if there is only one Arm, it remains one, if there are two Arms, two remain). The new value is the first valid Arm index after an increment. Pressing SHIFT+ARM increases in circular mode the index of the synchronised Arm. It is the only combination of keys able to change the quantity of Arms in the Status bar. It always passes to two Arms when there is only one Arm in the Status bar. It returns to one Arm when the index of the second Arm becomes equal to the first Arm.

  In DRIVE-ON - pressing ARM has two effects:

  - if DRIVE-ON has taken place with only 1 Arm selected ($TP_SYNC_ARM[2]=0), the Arm index is incremented, as for DRIVE-OFF;
  - if DRIVE-ON has taken place with 2 Arms selected, the selection is inverted for the Arm that will be moved with jog keys.

  For example, if there are 3 Arms and the current situation is Arm: 2<1, where Arm 2 is that moved by jog keys, pressing ARM, the index is NOT incremented (it does NOT change to 3), but the situation becomes Arm: 1<2 , i.e., the Arm moved by jog keys is now number 1. The first number displayed is always the Arm moved by jog keys. SHIFT+ARM cannot be pressed.

  A **LONG PRESSING** of the **ARM** key cyclically displays the possible indexes of the main Arm: when the required index is reached, releasing the key makes this choice definite. While the key is pressed, the index is displayed in brackets. When the final choice is made, the brackets disappear.

- (***TP4i***) STEP key- Sets continuous running mode (STEP DISABLED) on the active current arm move program.
- (***TP4i***) REC key - in editing environment, it inserts a move instruction, the declaration of the corresponding position variable to which the current arm position is assigned.

TP4i

TP4i

WiTP

Other colours keys

Depending on the available Teach Pendant (either TP4i or WiTP), the keys layout is slightly different: see the figures on the left. The keys functioning does not depend on the TP model.

– RESET (white) - press to reset the ALARM state; if the cause that generated the alarm concerns safety devices (e.g.: TP mushroom button, external mushroom button, automatic line barrier, etc.), the alarm is not deleted until the cause has been removed.
– START (green) - in PROGR mode it is used to execute movements (from editing/debug environment or Execute command) for all the time it is kept pressed. In LOCAL status it starts the motion program that is in READY state, waiting for this key.
– HOLD (yellow) - pressing this key stops all HOLDable programs and the motion for all Arms. The next time it is pressed the HOLD status is removed.

Alphanumeric keypad - The alphanumeric keypad operates in the same way as the most widely used standard keypads for mobile phones. Furthermore:

– The bottom right key (on the right of '0' key) is used to set the keypad mode. Each time this key is pressed the mode changes, in sequence, between
  • alphabetic low case (‘abc’)
  • alphabetic high case (‘ABC’)
  • numerical (‘123’)
  • fixed number ('123*'); contextual information: if the context requires it, the alphanumeric keypad is set in numeric mode and it is not possible to pass over to the another mode.
– The current mode is displayed on the Status bar.

TP4i

WiTP

– Special characters are activated by pressing '1' key; the only special characters that can be directly entered are '-' symbol (bottom left key) and '.' symbol (bottom right key). When in “high case alphabet” or “low case alphabet” mode, pressing key “1” activates a virtual keyboard for facilitated entry of characters and symbols.

Using the special characters of the virtual keyboard

To choose the required symbol, move the cursor inside the suitable group, by means of the cursor keys, and press ENTER to confirm.

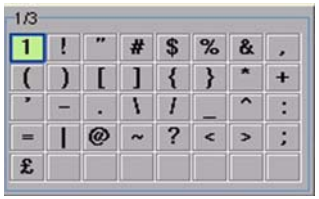
To enter the ‘space’ character, select any empty soft key (i.e. that does not correspond to a symbol): the last 6 in Window 1, the last 4 in both Window 2 and in Window 3.

To pass from one window to the next or to the previous one, use the Page Up and Page Down keys (see Navigation keys). To exit, act in one of the following ways:

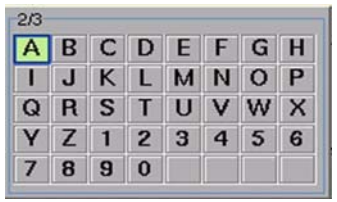
– press ESC
– press key ‘1’ again.

Three groups of characters are available:
– special characters (Window 1)
– high case alphabet + numbers (Window 2)
– low case alphabet + numbers (Window 3)

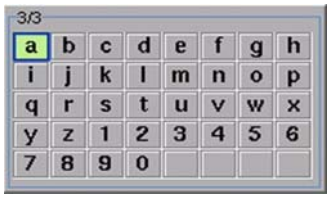
Window 1



Window 2



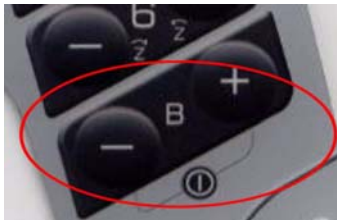
Window 3

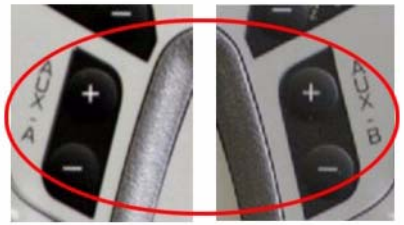


Keys to switch on the WiTP wireless

If the Teach Pendant is the WiTP wireless type, to switch on, press the AUX B+ and AUX B- keys simultaneously, for at least 2 seconds.
However, these keys can be released once all the Teach Pendant LEDs are alight.

It is recommended, even if not compulsory, to switch the Teach Pendant on, while it is in the docking station.





Keys to restart the Teach Pendant

If the Teach Pendant becomes locked in a condition from which autonomous re-enabling is not possible, or in any case a restart is necessary for other reasons, press simultaneously the four pushbuttons AUX A+, AUX A-, AUX B+ and AUX B-.

Warning! The **Restart with WiTP wireless** is different; after restarting, an **Emergency Unpairing** operation is needed, followed by the **Pairing procedure between WiTP wireless Teach Pendant and C4G Control Unit**.

6.2 Pushbuttons

The pushbuttons are divided into:

– Front pushbuttons - On the front face of the Teach Pendant, at the top, there is the Stop pushbutton; the operating modes are as follows:
  • activate by pressing
  • release by screwing (clockwise).

– Enabling Device - a full description has already been given before (Enabling Device operating mode).

TP4i

WiTP

6.3 LEDs

There are 4 LEDs on the Teach Pendant, located on the upper part of the device.

Their meaning (from left to right) is the following:

– Yellow LED - reserved

– Yellow LED - lit means that no NON-HOLDABLE programs are active

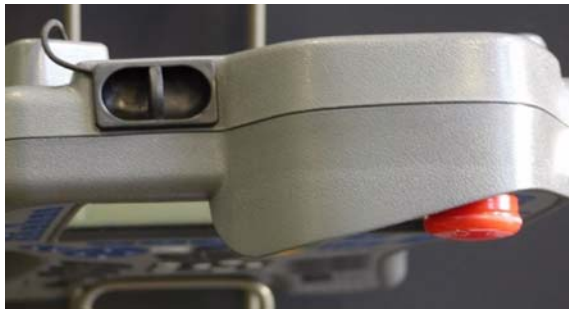
– Green LED - when ON, it means DRIVE ON. During the DRIVE ON procedure, this LED flashes. The light becomes fixed when the procedure has finished

– Flashing Red LED - Any type of alarm (except Warning type indications).

6.4 USB port

In the upper part of the Teach Pendant there is a USB port: in the pictures on the left, the USB port both with protection cover and without protection cover is shown.

It is recommended to use the USB port to connect a storage device such as a Disk-on-Key one.



7 Teach Pendant User Interface - basic information

The display of TP4i Teach Pendant is a graphic 6.4" TFT colour display; resolution 640x480 pixel.

After a few minutes that it is not used, the display switches off; to switch it on again just press any key.

The User Interface screen can be divided into the following areas:

- Status bar
- Messages bar
- Left Menu
- Right Menu
- Bottom Menu
- User Interface Pages.

8 Status bar

The status bar gives information about the system state; there are 7 alphanumeric fields on a single line that have the following meanings:

– The First Field indicates the system state: Auto-T (optional), Progr, Local, Remote.

– The Second Field gives further information about the system state, Holdable programs and movement while programming.
  - With the modal selector switch in AUTO, REMOTE and T2 (optional) positions, that is, in the Automatic positions, its meaning is as follows:

    **** - indicates that no holdable programs are running
    RUN - indicates there is at least one holdable program running

  - With the modal selector on T1, that is in programming state, the meaning is as follows:

    **** - indicates that no movements are in progress
    JOG, FORW, BACK - indicates that a manual movement is in progress, forward, backward, respectively

  - The meaning of HOLD and ALARM does not depend on the state of the selector switch:

    HOLD - indicates that the HOLD pushbutton is pressed and therefore the execution of holdable programs and all arm movements are stopped; it is shown on a YELLOW background
    ALARM - indicates that the system is in alarm state (N.B.: warning messages do not change the system state) and it is shown on a RED background

– The Third Field indicates the current value of the override percentage; if this field displays the character 'I' (e.g.: '80%(I)'), it means that the system is in incremental movement mode.

– The Fourth Field displays the number of the current Arm. In a Multiarm system with synchronised motion enabled, this field contains first the number of the main Arm, and then that of the SyncArm; example: A:2<1 means that Arm 2 is the main Arm and Arm 1 is the synchronised Arm). If there are Auxiliary axes, these refer to the first (or only) Arm indicated in the field; the sub-field marked with J indicates which auxiliary axis is assigned for jog movement, to key Aux-A and to key Aux-B respectively. If no axis is associated, a dash ('-') is displayed.

– The Fifth Field displays the current management mode of the coordinates:
  - TPM Jog mode: Base, Tool, Joint, Uframe
  - WRIST mode: Wr-Base, Wr-Tool, Wr-Uframe.

– The Sixth Field contains three sub-fields with the following information:
  - arm status:

    '-' arm correctly configured and ready for moving
    'turn' arm requires the turn-set operation
    'no cal' arm not calibrated
    'simu' simulated arm
    'dis' arm disabled
    coop' arm-arm cooperation ('An, Am') or axis-arm ('Ji, An')
  - state of both Enabling Device and Drives

    'E-' Enabling device not pressed
    'ED' with green background - Enabling Device pressed
    'ED' with red background - Error (e.g. both Enabling Devices pressed)
    'ON' - indicates the state of the drives (Drive ON)
  - presence of Latched alarms:

    'A' indicates that a latched alarm is present

  The background of this field may be:
  - green - arm properly configured and ready to move, and Enabling Device ok
  - yellow - Enabling Device ok and arm simulated or disabled
  - red - error on Enabling device or arm not calibrated or requires turn set operation

– The Seventh Field contains two sub-fields:
  - Alphanumeric keypad operating mode: abc (low case alphabet), ABC (high case alphabet), 123 (numbers), 123* (fixed numbers)
  - State of SHIFT key: 'S' indicates that the SHIFT key is pressed.

– If the Teach Pendant is WiTP wireless, the status bar contains an Eighth Field to display the current state of the Teach Pendant battery (see eighth field in the next figure):

9 Messages bar

The displayed text includes the following information:
- signal code of 5 digits, which identifies each message in an unique way
- error/alarm severity level
- error/alarm message description string.
- Displays the last occurred alarm/error message.

The background may be:
- green: if there is no message, this means there is no alarm/error in progress; if instead there is a message with an orange text, this means that an INFORMATIONAL message is active. The system is not in ALARM condition.
- blue: WARNING message.
- orange: this error is generated by the application process. The system is not in ALARM condition; the message only disappears when it is reset. From the Alarm Page, use the "down arrow" key to select the message then press Confirm (F1) in the Central Menu.
- yellow: this is a PROC message
- grey: the text is black. This is a Latched message and requires an explicit consent from the user. To do so, use the "down arrow" from Alarm Page, sub-page Latched, to select the message, if necessary, remove the cause of the alarm, then press Confirm (F1).
- red: this is an ERROR/FATAL message. The system is in ALARM state.

10 Left Menu

L1

L2

L3

L4

L5

L6

It is a group of softkeys that allows access to the User Interface Pages (the figures on the left show the pre-defined softkey icons).The corresponding keys are indicated with the names L1..L6, in sequence.

General criteria for use of the Left Menu:

- if the corresponding User Page is not yet active, when the key is pressed it will be immediately displayed;
- if the corresponding User page is already active, when the key is pressed it will close, and the Home page will be shown on the display.

The softkey corresponding to the active page is highlighted by a blue background. If there is no softkey highlighted, this means that the page currently displayed is the Home Page.

If the softkey background is yellow, this means that a command associated to this softkey is being run. It will remain yellow until the operation has been completed.

To view all the soft keys available in the Left or the Right Menu it is necessary to use:

- **SHIFT+MORE (left keys for the Left Menu, right keys for the Right Menu), to view the PREVIOUS soft keys**
- **MORE (left key for the Left Menu, right key for the Right Menu) to view the NEXT soft keys.**

11 Right Menu

Local	Remote	Progr	Auto-T	Multiarm		Description
					R1 R2 R3 R4 R5 R6	This menu is dedicated to the technological commands such as the opening or closing of the grippers and the welding parameter hardware settings The corresponding keys are called R1..R6, that correspond to softkeys U1..U4 which are keys that can be programmed by the user, plus another two keys that depend on the state of the system: – When the system is in Local or Remote mode, key R5 is always for the Drives on/off, as described below: • Local - R5 key allows Drive ON and Drive OFF commands (toggle key); the corresponding softkey is displayed with a green background if the drives are on, otherwise with a grey background. • Remote - R5 is for Drive OFF command only (Drive ON comes from the PLC); the background of the corresponding softkey is grey for Drive OFF and green for Drive ON. – Progr - R5 and R6 are for T1 and T2 keys (that control tools 1 and 2, to execute switching between OPEN and CLOSE). – Auto-T (optional) - R5 and R6 are not associated to any function. When one of such keys is pressed, it activates the execution of the associated command and displays the state (if the softkey is highlighted, the corresponding command is active; otherwise the command is not active). If the softkey background is yellow, this means that the associated command is in progress. It remains yellow until the execution has been completed. If the system is Multiarm and the Teach Pendant is TP4i, one of the Right Menu possible configurations is the one shown in **Multiarm column, here beside**. This softkey allows to change the current Arm.

To view all the soft keys available in the Left or the Right Menu it is necessary to use:
- **SHIFT+MORE (left keys for the Left Menu, right keys for the Right Menu), to view the PREVIOUS soft keys**
- **MORE (left key for the Left Menu, right key for the Right Menu) to view the NEXT soft keys.**

12 Bottom Menu

Fig. 1 - Example - Bottom Menu of the Home Page

This menu is a group of 6 function keys, F1..F6, contextual to the selected item (current state and page, selected field, etc.). The shown functions correspond to the possible actions on the selected item.

Usually when one of the keys F1..F6 is pressed the associated function is activated directly. If the softkey background is yellow, this means that the associated command is in progress. It remains yellow until the execution has been completed.

However there are some softkeys that are associated to a pop-up sub-menu: this is indicated by a small triangle on the top left corner of the icon. For example, this is the case of the softkey to choose the language, F1 on the Home Page. See the figure here beside.

When pop-up sub-menus are opened, the first item is always pre-set. The choice of an item can be made as follows:

– by pressing the associated digit of the required item on the Alphanumeric keypad, or
– by moving in the sub-menu using cursor keys and confirming the choice with ENTER key.

It is also possible that a softkey corresponds to a function that foresees optional settings: items of this type are identified by a small icon representing a clock, in the top right corner of the softkey (e.g. key Login (F2) on Home Page). A short pressure on the key associated to one of these commands causes the function to be executed according to a default; a longer pressure causes the opening of a pop-up sub-menu for the choice of the options

The pop-up window closes at any time, either because a choice has been confirmed or the ESC key has been pressed (exit that cancels any action).

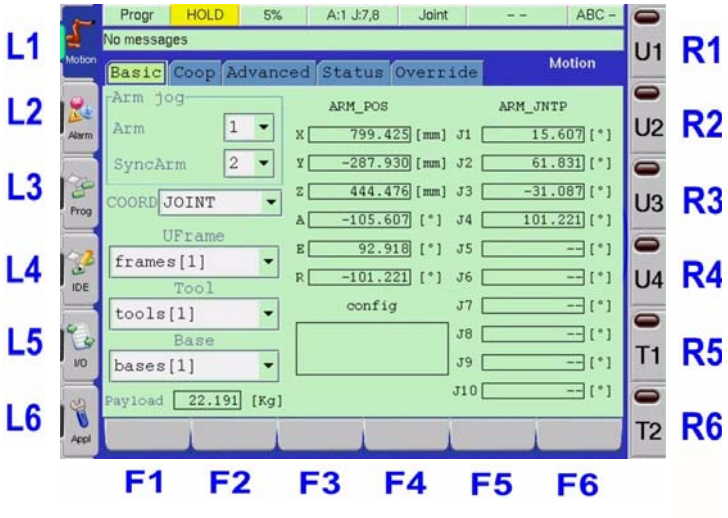
13 User Interface Pages

These are the User Interface Pages on the Teach Pendant.

The active Page has a narrow blue frame that graphically links to its name shown in the top right corner; the corresponding softkey, in the Left Menu, is highlighted. See the example in the figure here beside (motion environment - **Motion Page**)

To activate a User Page, press the corresponding softkey in the Left Menu.

Each Page refers to specific functions in a precise ENVIRONMENT (for example Motion environment, I/O environment, etc.).



The available User Pages are as follows:

- **Home Page** - When no User Page is active, a default screen is displayed by the Teach Pendant; this screen is called Home Page. It is associated with no key of the Left Menu, however it turns visible whenever a User Page is closed.
- **Motion Page** - This User Page manages information regarding the robot motion environment and allows the user to display and change it.
- **Alarm Page** - This User Page allows management of information linked to alarms/errors that have occurred in the system.
- **Prog Page** - Purpose of this User Page is to handle the programs.
- **IDE Page** - The IDE environment (Integrated Development Environment) is used to develop motion programs (programs with HOLDABLE and EZ attributes). In IDE environment the position teaching functionality is integrated.
- **I/O Page** - This User Page allows to handle the I/O available in the Control Unit C4G:
- **Appl Page** - This page is the interface of the application package installed on the C4G Controller Unit.
- **Files Page** - The User page of File Manager (Files) allows to carry out all the operations concerning the files: copy, cancel, create and remove folders, etc.
- **Data Page** - The User Data page of the Teach Pendant allows to look at and modify all the data tables stored in the Control Unit C4G.
- **Setup page** - The Setup page is used to read and change the system settings.
- **Service Page** - This User Page is the starting point for the service operations.
- **TP-INT Page** - This page emulates the user interface of the WinC4G terminal on the Teach Pendant.

14 System power on and shut down

- Power on procedure
- Shut down procedure.

14.1 Power on procedure

a. close the door (or doors) of the Robot Controller Cabinet.

b. Check that these cables are connected:
   - power supply cable
   - cables X10 and X60 to the robot
   - safety devices cable X30.

Wait at least 20 seconds after a power-off before re-powering the control unit.

c. Turn on the electric power supply by turning the main switch to ON.

d. If it is wished to use the Teach Pendant, the steps to be followed depend on the type of Teach Pendant installed on the system:

d.1 power-on the Teach Pendant (if it is off), pressing keys AUX B- and AUX B+ simultaneously - See Keys to switch on the WiTP wireless

d.2 place the WiTP on the docking station (if not already there) and wait for the information "system ready for pairing"

d.3 carry out the pairing with the Control Unit

e. if it is expected to execute robot movements, switch the drives on and press START button. See par. 16 Execution of operations requiring the Robot motion on page 1-45.

14.2 Shut down procedure

To shut the system down, it is enough to simply move the main switch to OFF. It is in any case strongly recommended to follow the software shutdown procedure, to avoid unnecessary waste of RPU buffer battery cycles that would reduce the life of the battery. Furthermore, in the case of a system with wireless Teach Pendant, it is NECESSARY to use the software shutdown to close in consistent situations.

The full procedure is as follows:

a. set the system in DRIVE OFF

b. issue the Software Shut down command (either form TP or from PC or from PDL2 program).

b.1 If the TP is a wired TP:

b.1.1 after approx 5 seconds the Teach Pendant display is cleared. Wait at least 10 seconds

b.1.2 move the main switch to OFF. End of procedure.

b.2 If the TP is a wireless one:

b.2.1 place the WiTP on the docking station

b.2.2 press the pairing/unpairing pushbutton. One of the following scenarios might occur:

b.2.2.1 if all proceeds correctly, the system shuts down the WiTP; therefore the user can use the main switch to shut down the C4G Control Unit;

b.2.2.2 the system detects a fault BEFORE the pairing/unpairing pushbutton has been pressed - the C4G Control Unit does NOT prepare for shutdown and the WiTP is NOT shutdown. The shutdown command is aborted and the user is notified by a specific message. The situation is the same as it was before the shutdown request;

b.2.2.3 the system detects a fault AFTER the pairing/unpairing pushbutton has been pressed - the C4G Control Unit shutdown is suspended, but the WiTP is unpaired and off. It is, in any case a consistent situation. The user can act on the main switch to shut down the C4G Control Unit.

15 Access to the Control (LOGIN/LOGOUT)

To access the Control Unit, regardless of the device from which dialogue with the controller is desired (Teach Pendant or WinC4G program from PC), a Login is necessary, otherwise the only commands available are for viewing.

To terminate access to the Control Unit, so that unauthorised personnel cannot enter, it is necessary to execute a Logout.

15.1 Login

A Username and a Password must always be specified, to be acknowledged by the Controller to be interfaced; these must be defined and saved beforehand in the Controller database.

There are five types of user predefined and recognised by the control unit:

– Administrator **-** The only task of this user is to enter and/or delete the users in the data base (file .UDB) that gives access to the system; therefore many other commands are not enabled for the Administrator.

– Programmer **-** The programmer user is enabled, mainly, to execute the operations associated to the development, the verification and the setting up of the programs.

– Maintenance **-** The type of user to be identified is the integrator. This user has more potential than the programmer.

– Service **-** He/she also performs servicing and is a user enabled to execute operations connected to system updating, using commands for software loading and machine calibration.

– Technology - With the technology user access is allowed to some functions of the installed application, typical of that application. See the specific application manual for further information.

The C4G Control Unit is delivered to the customer with the following predefined users:

– programmer user - Username: pu Password: pu

– maintenance engineer user - Username: mu Password: mu

– administrator user - Username: admin Password: admin

If the predefined users (programmer or maintenance engineer) are not those required by the end customer, others can be defined. See User subpage in the Setup page.

The Login operation can be performed either by TP (Home Page) or by PC (SetLogin command from WinC4G).

15.2 Logout

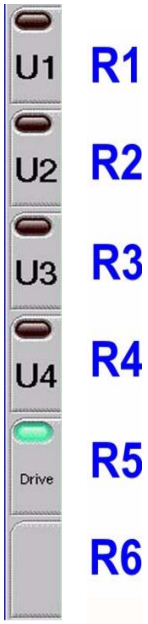
If on the TP, press long the **Login** (**F2**) softkey, from the Bottom Menu of the Home Page. If working on the PC (WinC4G), either press the “disconnect” key or issue the **SetLogin/L** command from the commands menu.

A function called Automatic Timed Logout is available. It has been implemented to prevent unauthorized personnel from executing potentially dangerous operations, during the system working cycle. The main purpose is to perform a Logout opertion after a certain interval of user inactivity on the TP.

16 Execution of operations requiring the Robot motion

When the modal selector switch is set on position T1 (or T2 - optional service), the programs can be developed using editor environment and the spots can be taken from the Teach Pendant moving the robot manually with the motion keys; the programs can be set up using the debug tools of the system. In programming mode, the execution of a move instruction requires that the operator presses the START key and the enable device on the Teach Pendant.

When the modal selector switch has been set on T1, or T2 or AUTO, the system is under the control of the operator. When the selector is set on REMOTE, the system is under remote control (for example from PLC).



Before any operation can be executed that requires movement, the drives must be powered:

– if the modal selector switch is in either T1 or T2 (optional) position, press in the intermediate position the Teach Pendant Enabling Device, to power ON the drives; tho switch them OFF and activate brakes on all axes controlled by the Control Unit, just release the Teach Pendant Enabling Device,

– if the modal selector switch is in AUTO position, press the R5 softkey (Teach Pendant right menu - it means DRIVE ON when in AUTO state), to power ON the drives; to switch them OFF and activate brakes on all axes controlled by the Control Unit, press the R5 softkey again (Teach Pendant right menu - now it means DRIVE OFF).

– if the modal selector switch is in REMOTE position, DRIVEs ON and OFF are remote controlled.

17 Turn-set and Calibration

17.1 Terminology used

– TRANSDUCER: There are two types of position transducers: encoder and resolver.
– NUMBER OF TRANSDUCER TURNS: during the robot axis movement, the transducer may make several turns; the number of turns is initialised through the calibration or the turn-set.
– AXIS VALUE: the value of an axis contains all the information needed to determine the exact position of an axis in space;
– VALUE RECONSTRUCTION: when the Control Unit is powered on, the system software, among the various initialisations, reconstructs the value of the robot axes.
– CALIBRATION POSITION: a pre-set position that has been checked using specific equipment (dial gauges, supports, calibration fixtures). The calibration position is a reference position in the robot working space that serves to initialise the value of each axis.
– CALIBRATION CONSTANTS: the calibration constant is the difference between the datum read by the transducer and the nominal position of the robot axis that the transducer should assume in that particular position of the robot axis. In fact, since the positioning of the transducer as to the robot joint is casual, (because it depends on how the transducer has been mounted), it is necessary to correct the actual position of the transducer according to the nominal position required by the robot axis.
– CALIBRATION ASCII FILE: the calibration file UD:\SYS\<$SYS_ID>_CAL<num_arm>.PDL (where $SYS_ID indicates the system identification, for example NH4_001) is an ASCII file with syntax of a PDL2 file, where the calibration constants ($CAL_DATA[n]) and other typical data of the robot are stored.
– NVRAM: the memory used to save the characteristic information of the robot associated to the Control Unit, the calibration constants and the length of the levers. It is the partition, on the SMP+ board, for the motion process.

17.2 Basic concepts

17.2.1 Calibration

The purpose of the calibration procedure is to establish the position of a robot axis referring it to an ideal robot. This makes it possible to initialise the values of the robot axes and to make the position variables used in the robot programs universal.

During the calibration procedure, when the desired axis is in the calibration position, two values are stored:

– the deviation, inside a transducer turn, between the value of the actual position and that of the axis nominal position,

– the number of transducer turns.

The notches on the individual axes make it possible to execute future turn-set operations on a robot that has already been installed. The recovery of the calibration (executed by COMAU), if necessary, is to be executed when first putting the robot into operation.

Subsequently, the calibration does not need to be executed again, unless there is a mechanical failure that involves the replacement of a component of the kinematic chain, or in the case of impacts that damage the robot structure.
In situations in which calibration data are lost, the calibration procedure IS NOT required. Use the Load (CARL) command from Ret. Mem. (Mem.Ret.) of the Setup page - Arm subpage, to recover the data from NVRAM. If the user wishes to recover them from the calibration file, use Load File. Once calibration data have been recovered, execute the Turn-set to initialize again the transducer turns number.

17.2.1.1 System calibration

To initialise the robot axis values in the system calibration position (calibration position predefined by COMAU Robotics - $CAL_SYS). To determine the correct calibration position, special equipment has to be used (dial gauges, supports, etc.) to determine with the necessary precision the position of each individual axis.

17.2.1.2 User calibration

User calibration defines a new calibration position that is different to that of the system. This type of calibration (commonly called out-of-range calibration) can be used when the system position is difficult to reach once the robot is inserted in the final application, and therefore it becomes necessary to define a different calibration position, called user calibration position ($CAL_USER). It is the responsibility of the user to provide the appropriate instruments and to check the correct positioning of the robot in any user re-calibrations, especially regarding the arrangement of the locating notches.

17.2.2 Turn-set

The purpose of the turn-set is to update the number of transducer turns only, should it occur that the when switched on again, the Control Unit has lost this value. The operation consists in bringing the axis involved to the calibration position, using the locating notches, and giving the required command. No special equipment is needed, because the only value initialised is the number of turns of the transducer. The turn-set operation is required when

– there has been axis movement with the control off (for example when the error 59411 SAX - movement after shut-down) is displayed.

– events take place that cause the loss of the number of turns only, and therefore do not require the execution of the calibration procedure. On the Teach Pendant status window or on the PV video the text Ar:TURN is displayed.

17.2.3 Calibration Position

An example of calibration position for the SMART SiX robot, defined when all the axes notches are aligned, is shown in the figure on the right:

1. axis 1 calibration index
2. axis 2 calibration index
3. axis 3 calibration index
4. axis 4 calibration index
5. axis 5 calibration index
6. axis 6 calibration index

Joint position when calibrated ($CAL_SYS)

Axis 1	Axis 2	Axis 3	Axis 4	Axis 5	Axis 6
0°	0°	-90°	0°	+90°	0°

17.3 Procedure

- Turn-set procedure
- Calibration procedure.

17.3.1 Turn-set procedure

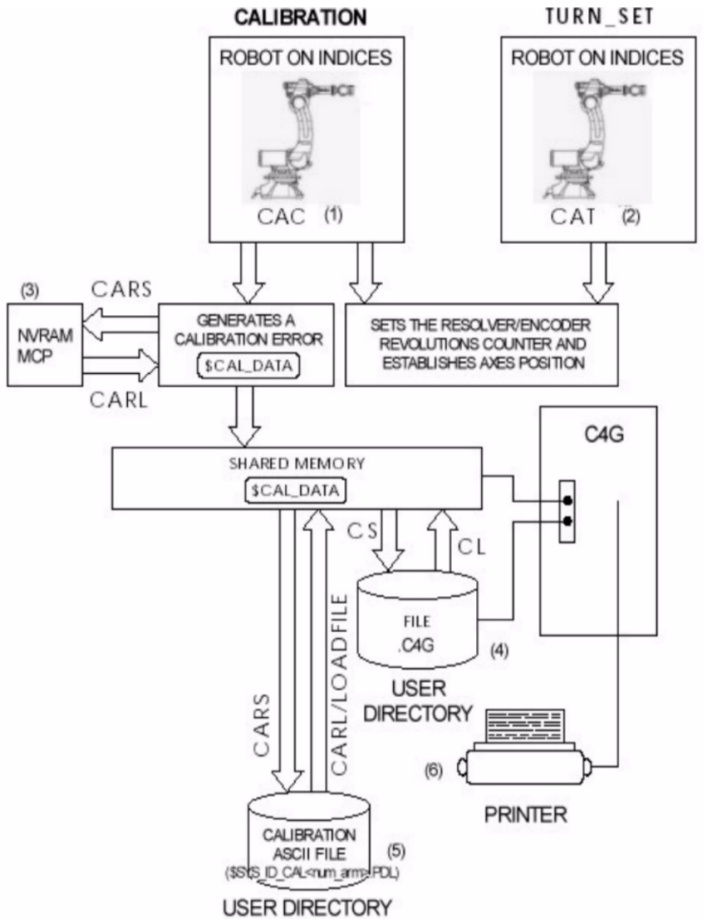
This procedure has to be executed if the Control Unit knows the calibration constants, but has lost the information regarding the number of transducer turns, which could, for instance, be caused by an electrical disconnection of the motors.

a. Bring the robot axes on the locating notches to the most precise position possible that can be obtained by sight: the system will warn the operator with error 59409 SAX: joints position not accurate enough and then will display error 59421 SAX: a positive recording is required and/or error 59422 SAX: a negative recording is required to reach the correct position.

b. If the arm number is not specified, the default arm is used, whereas the axis number must always be specified. To indicate all the arm axes, enter an asterisk (*).

c. With the robot positioned correctly set the CAT (Configure Arm Turn-set) instruction. If working on the TP, use Calib subpage of **Setup page - Arm**.

After the CAT instruction, the system re-calculates the correct number of transducer turns according to the calibration position ($CAL_SYS). The operation is only to be executed on the axis that has lost the count; if there are several axes in this situation it is better to run the turn-set of one axis at a time. In particular, if there are axes with mechanical positioning influences it is necessary to operate first on the influencing axis and then on the influenced axis (follow the sequence axes 4,5,6).



1. - CALIBRATION command
2. - TURN-SET command

Saving the calibration constants
3. - NVRAM
4. - UD:\SYS in file.C4G
5. - UD:\SYS in the calibration file ($BOARD_DATA[1].SYS_ID_CAL.PDL)
6. - Hard copy

17.3.2 Calibration procedure

Depending on the desired accuracy level, the following Calibration procedures are available:

– Calibration procedure using the locating notches

– Calibration procedure using fixtures.

The degree of precision obtained when making the calibration influences the final precision of the robot positioning

If the calibration is not very accurate, the Tool Center Point and the fixtures installed could reach technological points of the work cycle with less precision than that obtained with the previous calibration.

The Calib (CAC) command provides the following options:

– **Learn (CAC/L)** - Learns the actual position as the user calibration position ($CAL_USER). The learning procedure is executed on all the selected axes.

– **NoSave (CAC/N)** - No automatic saving is performed of the file .C4G, ASCII calibration file <$BOARD_DATA[1].SYS_ID>_CAL<num_arm>.PDL and calibration data in the NVRAM memory.

– **User (CAC/U)** - For the calibration position it uses that of the user obtained or assigned beforehand ($CAL_USER).

– **Learn NoSave (CAC/LN)** - This is a combination of the previous options Learn (CAC/L) and NoSave (CAC/N).

– **User NoSave (CAC/UN)** - This is a combination of the previous options User (CAC/U) and NoSave (CAC/N).

17.3.2.1 Calibration procedure using the locating notches

The calibration by means of the notches enables a quick but improper calibration that has a limited precision and may not restore the robot movement precision required by the specific application.

Calibration using the notches consists in bringing the robot axes onto the calibration notches, aligning them by sight without using specific fixtures and executing the calibration commands axis by axis as specified in the following par. 17.3.2.2 Calibration procedure using fixtures on page 1-51.

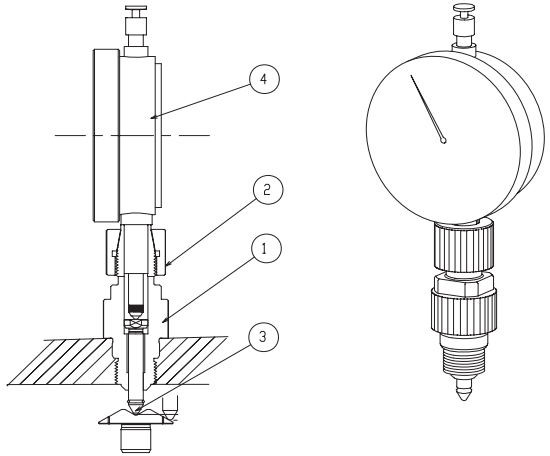
17.3.2.2 Calibration procedure using fixtures

Calibration using fixtures has to be executed after mechanical disassembly operations that change the robot geometry, or if it is necessary to make a precise check on the calibration position.

The 1-2-3 axes calibration requires an appropriate dial indicator holder (see the figure on the right) screwed onto the pre-engineered seat on each axis of the robot: the dial indicator is used to measure the minimum point with reference to the fixed index consisting of a V-shaped notch: the position of the robot at the lowest reading is the calibration position. For the calibration of axes 4-5-6 the dial gauge holder is mounted on the surfaces provided on the forearm and on the wrist, using a specific interfacing support (see the figure on the right).

The calibration has to be carried out in progressive sequence, starting from axis 1 and continuing with axes 2, 3, 4, 5 and 6. The following rules that arise from the axis positioning mechanical influences are also to be kept in mind.

– If axis 4 is calibrated, also axes 5 and 6 have to be recalibrated.
– If it is necessary to calibrate axis 5, also axis 6 has to be recalibrated.
– The calibration of a single wrist axis (4, 5 or 6) requires the other wrist axes, not involved in the calibration, to be positioned on the calibration notches.



1 Dial indicator holder
2 Cone-shaped locknut
3 Sensing head
4 Dial indicator

Procedure

a. Activate programming mode (St: PROGR) on the Control Unit.

b. Press the Enabling Device on the Teach Pendant, to power on the drives.

When the enable device on the Teach Pendant is released (DRIVE OFF) slight axis movement may be caused, generated by the gravity of the applied loads; these become more evident if the HOLD TO RUN functions are executed in quick succession.
Following every HOLD TO RUN of the system, check on the Teach Pendant that the axes return to the correct calibration position. If the axes do not return to the correct position, the calibration procedure will have to be started again from the beginning.

c. Remove the protection guards from the threaded seats for dial gauge holder fixture attachment and from the calibration locating notches. Execute the below listed steps for each axis.

The robot calibration has to be executed with no loads applied to the axes (tool, gripper, etc.) except axis 1.

d. Using the Teach Pendant move the robot axes to calibration position (see Fig. 7.1 - Robot calibration position) so that the moving index coincides with the fixed index.

e. Fit the dial gauge holder fixture in the seat then screw the dial gauge into the fixture to pre-load the dial gauge pointer by a few millimeters (approx. 2.5 mm).

f. Slowly, (not more than 5% - GEN-OVR < 5%), turn the axis to be calibrated moving from negative direction to positive, until the dial gauge indicates the minimum point.

g. On the Teach Pendant keyboard, select the Calib (CAC) command from the Arm - Calib subpage

h. Select the arm for calibration and confirm with ENTER.

i. Select the axis for calibration and confirm with ENTER.

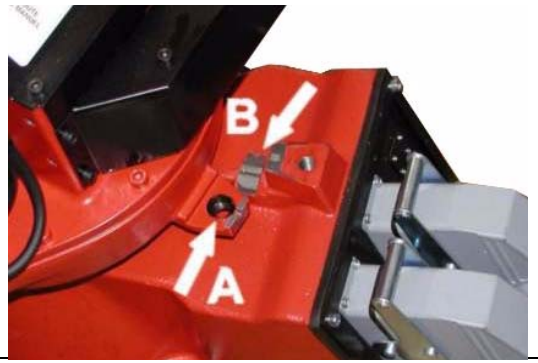
j. The calibration constants are stored in the variable $ARM_DATA[num_arm].CAL_DATA, in the calibration ASCII file, in file.C4G, in NVRAM.

Calibration procedure for each single axis.

- Axis 1 Calibration
- Axis 2 Calibration
- Axis 3 Calibration
- Axis 4 Calibration
- Axis 5 Calibration
- Axis 6 Calibration

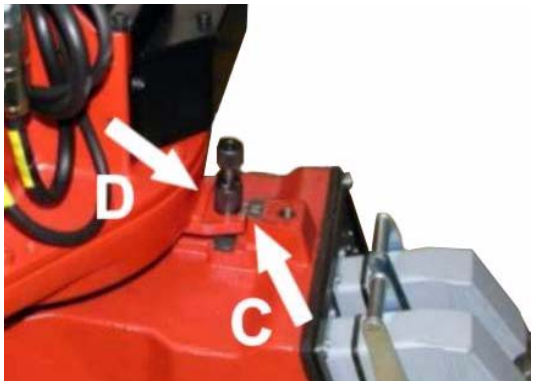
Axis 1 Calibration

a. Remove the protections from the reference index (B) and seat (A) for the attachment of the dial gauge holder fixture.



b. Align by sight the calibration notches (C).

c. Fit the dial gauge holder fixture (D).



d. Finding the minimum reading on the dial indicator, which is the calibration point.

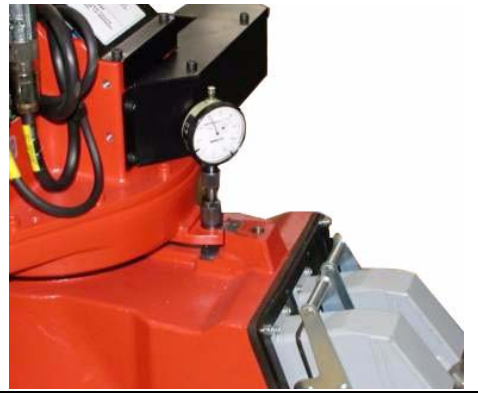
e. If the minimum point has been passed, return to the start position and repeat the measurement, always moving the robot axis from negative direction to positive

f. On the Setup Page, select the Calib (CAC) command from the Arm - Calib subpage.

g. Select the arm for calibration and confirm with ENTER.

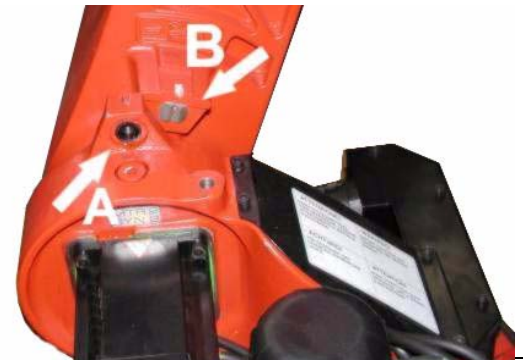
h. Select the axis for calibration and confirm with ENTER.

i. Remove the fixture and refit the guards on the calibration notches.



Axis 2 Calibration

a. Remove the protection from the calibration locating notch (B) and from the seat (A) for attachment of the dial gauge holder fixture.



b. Align by sight the calibration locating notches (C).

c. Assemble the dial gauge holder fixture (D).

d. Find the minimum point on the dial gauge that corresponds to the calibration point.

e. If the minimum point has been passed, return to the start position and repeat the measurement always moving the robot axes from negative direction to positive.

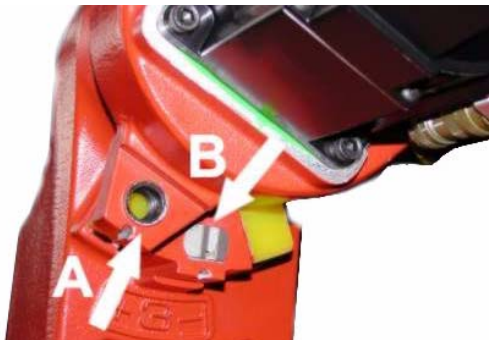
f. On the Setup Page, select the Calib (CAC) command from the Arm - Calib subpage.

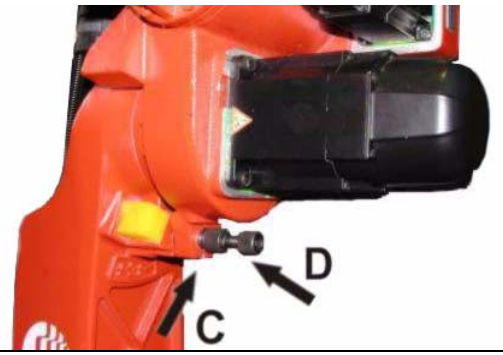
g. Select the arm for calibration and confirm with ENTER.

h. Select the axis for calibration and confirm with ENTER.

i. Remove the fixture and refit the guards on the calibration notches.

<table>
<tr>
<td><u>Axis 3 Calibration</u>

a. Remove the protection from the calibration reference index (B) and the seat (A) to attach the dial gauge holder fixture.</td>
<td> 
</td>
</tr>
<tr>
<td>b. Align by sight the calibration locating notches (C).

c. Fit the dial gauge holder (D) fixture.</td>
<td> 
</td>
</tr>
<tr>
<td>d. Fit the dial gauge on the fixture.

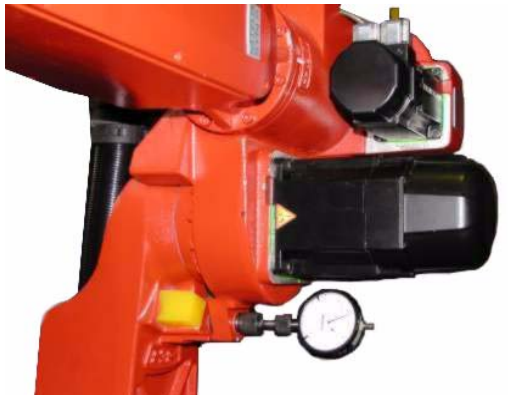
e. Find the minimum point on the dial gauge that corresponds to the calibration point.

f. If the minimum point is passed, return to the start position and repeat the measurement always moving the robot axis in the direction from negative to positive

g. On the Setup Page, select the Calib (CAC) command from the Arm - Calib subpage.

h. Select the arm for calibration and confirm with ENTER.

i. Select the axis for calibration and confirm with ENTER.

j. Remove the fixture and refit the guards on the calibration notches.</td>
<td>  </td>
</tr>
</table>

Axis 4 Calibration

a. Removal of the protection from the calibration location index (A).

b. Align by sight the calibration locating notches (B).

c. Assembly of support (C) for calibration fixture. Position the support and fasten it with an M5 socket head cap screw and two parallel pins dia. 6x20.

d. Check the perfect planarity of the block with the resting surface.

If necessary, use a plastic hammer when assembling the support

e. Dial gauge holder support (D) assembly on the forearm surface provided	
f. Assembling the dial indicator on the holder. g. Finding the minimum reading on the dial indicator, which is the calibration point. If the minimum reading has been passed, return to the starting position and repeat the measurement always moving the robot axis in the negative to positive direction. h. On the Setup Page, select the Calib (CAC) command from the Arm - Calib subpage. i. Select the arm for calibration and confirm with ENTER. j. Select the axis for calibration and confirm with ENTER.	
k. Remove the dial gauge holder using a screw-type puller and refit the protections on the calibration notches	

Axis 5 Calibration

a. Removal of the calibration location index (A) protection

b. Align by sight the calibration locating notches (B)

c. Assembly of support (C) for the calibration fixture. Position the support and secure it with the two parallel pins dia. 6x20 and the M5x25 socket head cap screw

d. Check the perfect planarity of the block with the resting surface.

If necessary, use a plastic hammer when assembling the support

e. Assembling the dial indicator holder (D).

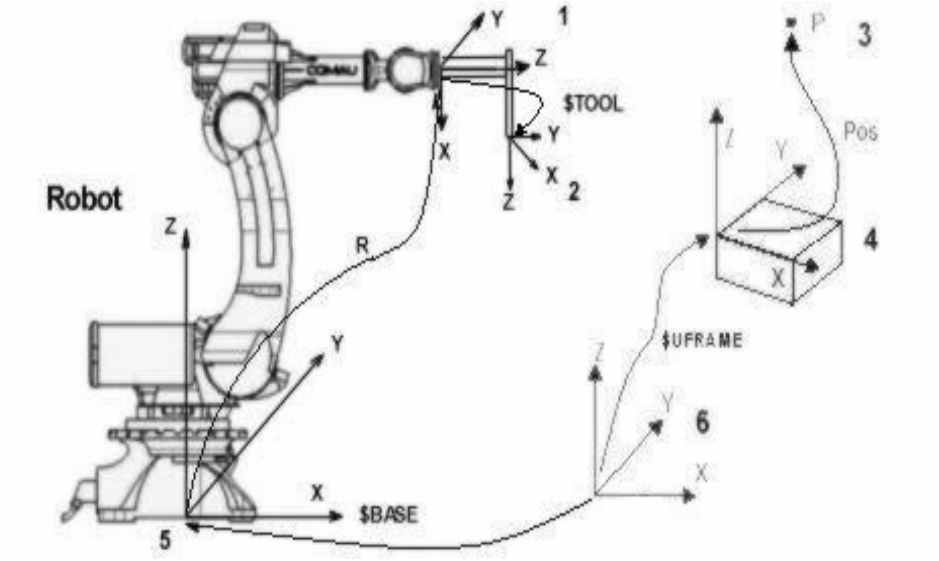
f. Assembling the dial indicator on the holder. g. Finding the minimum reading on the dial indicator, which is the calibration point. If the minimum reading has been passed, return to the starting position and repeat the measurement always moving the robot axis in the negative to positive direction. h. On the Setup Page, select the Calib (CAC) command from the Arm - Calib subpage. i. Select the arm for calibration and confirm with ENTER. j. Select the axis for calibration and confirm with ENTER.	
k. Remove the dial gauge holder using a screw-type puller l. Refit the previously removed protections on the calibration index	

Axis 6 Calibration a. Removal of the calibration location index (A) protection	
b. Align by sight the calibration locating notches (B)	
c. Assembly of dial gauge holder support (D) Position the support and secure it with the two parallel pins dia. 6x20 and the M5x25 socket head cap screw d. Make sure that it is perfectly level with the supporting surface	

e. Assembly of calibration fixture (E)	
f. Assembling the dial indicator on the holder. g. Finding the minimum reading on the dial indicator, which is the calibration point. If the minimum reading has been passed, return to the starting position and repeat the measurement always moving the robot axis in the negative to positive direction. h. On the Setup Page, select the Calib (CAC) command from the Arm - Calib subpage. i. Select the arm for calibration and confirm with ENTER. j. Select the axis for calibration and confirm with ENTER.	
k. Remove the dial gauge holder using a screw-type puller l. Refit the previously removed protections on the calibration indexes	

18 Robot motion in Programming mode

18.1 Reference frames



1. - Flange frame
2. - Tool frame
3. - Recorded position
4. - User frame
5. - Base frame
6. - World frame

Cartesian reference system

A Cartesian reference system, or reference set of three, is a geometrical concept to enable the representation of an object in space. For example, the corner of a table may be chosen as a reference system to represent the table. The same method can be applied for a book lying on a table, as for a weld gun mounted on the flange of a robot.

Co-ordinates conversion

A co-ordinates conversion describes the position of one reference system in relation to another. This is described as a POSITION variable. For example, if a table is located in a room, its position in relation to the room is indicated by POSITION p_table, that describes the co-ordinates conversion between the two reference systems. The co-ordinates conversion can also be used to calculate the position of an object in relation to different reference systems. For example, a book with a position in relation to the corner of the table is p_book and will have the position (p_table:p_book) in relation to the corner of the room. The sign (:) indicates the relevant position operation, and makes it possible to compose the effect of various co-ordinate conversions. For further information, see the PDL2 Programming Language Manual.

System reference frames

The Controller has three system variables ($BASE, $TOOL and $UFRAME) that permit the description of the main co-ordinate conversions. Before starting to explain these conversions, it is necessary to define some reference sets-of-three.

World frame	–	Workshop reference frame in relation to where the machines are positioned
Base frame	–	frame that indicates the robot base
User frame	–	frame that indicates the workpiece
Flange frame	–	frame that indicates the robot flange
TCP frame	–	frame that indicates the tool tip

The $TOOL variable describes the position of the TCP frame in relation to the flange; the $BASE variable describes the position of the base frame in relation to the world frame; finally, the $UFRAME variable describes the position of the workpiece in relation to the world frame.

The POS conversion indicates the recorded point P where the TCP will position when executing the program. It must be remembered that all the POSITIONS recorded are defined in relation to the user reference frame (defined by $UFRAME, with certain $BASE and $TOOL values).

Remember that, changing $TOOL or $BASE or $UFRAME, the same position (POS) corresponds to a different actual position of the robot!

Let’s now imagine a pen fitted on the flange of the robot that has to write the word COMAU on the table. The $BASE conversion defines the point where the robot base is located, the $TOOL movement indicates the pen and the $UFRAME movement indicates the position of the table.

18.2 Manual motion

The manual movement of the arm is necessary in certain circumstances, among which when learning (recording) the positions or during maintenance of the tool fitted on the arm. The black keys on the TP4i/WiTP Teach Pendant are used for manual motion. To be able to make the move it is necessary to have the system in programming status, i.e. with the modal selector switch in position T1, and the Enabling Device pressed.

Before starting to move, the movement mode and the speed should be selected.

From the Motion page of the Teach Pendant, Basic sub-page (COORD field), one of the following modes can be selected to move the arm:

– JOINT - joints mode. The ‘+/-’ keys are associated to each of the axes of the selected arm; the keys associated to any auxiliary axes present follow those of the arm (typically they are keys 7 and 8 (‘+/-’)). When one of the keys is pressed, the corresponding axis moves in the positive or the negative direction, according to the directions indicated on the plate on the arm.

– BASE - linear movement mode according to the tool reference x,y,z frame (or TCP frame). The first three '+/-' keys (on the left) are used for linear motion in the direction of the three axes of the world reference system); the next three '+/-' keys (on the right) are for the rotation of the tool around the same axes keeping the TCP position unchanged.. It must be remembered that the world frame is not defined directly by any system variable; in fact, it is the robot base that is represented in relation to the world by means of the $BASE variable.

– TOOL - linear movement mode according to the tool reference x,y,z frame (or TCP frame). The first three ‘+/-’ keys allow linear movement in the direction, of the three axes of the tool reference system (defined by the $TOOL variable); the next three ‘+/-’ keys are for the tool rotation around the same axes keeping the TCP position unchanged (tool working point).

– UFRAME - linear movement mode according to the user reference x,y,z frame (for example the frame that describes the workpiece). The first three ‘+/-’ keys allow linear movement in the direction of the three axes of the user reference system (defined by the $UFRAME variable); the next three ‘+/-’ keys are for the tool rotation around the same axes keeping the TCP position unchanged.

The speed of the manual motion can be selected with the +% and -% keys that act on a percentage value shown on the Teach Pendant status bar. This percentage value is called general override and does not only act on the manual movement speed, but on all types of movements, both in programming and in automatic mode.

The TCP movement speed, during manual movements, is always lower than the safety speed of 250 mm/s also in joints mode. In the Cartesian modes (Tool, Uframe, Base) the maximum speed that can be reached is limited by the system variable $JOG_SPD_OVR that usually has values equal to 50% (i.e. half the safety speed). This value can be changed to adapt the standard manual movement speed to the individual programming requirements.

Before moving in Cartesian mode (Tool, Uframe, Base) the correct definition should be checked of the reference systems, especially the declaration of the tool frame through the $TOOL variable. A wrong description of the tool causes errors in learning the points and does not keep the TCP position unchanged during orientation movements. A good method to check the correctness of $TOOL is to check that the TCP remains fixed while changing the orientation of the tool.

The procedure for arm manual movement of a robotic cell varies slightly according to the cell controller configuration. The following paragraphs describe the main details for each typical situation.

Basic Training Course - C4G USE AND PROGRAMMING - Second Day

- **TOOL and UFRAME definition**
  1 - TOOL and FRAME automatic calculation

- **Motion Control**
  2 - Types of movements overview
  3 - Motion Dynamics

- **Creazione dei Programmi**
  4 - Introduction to the IDE programming environment

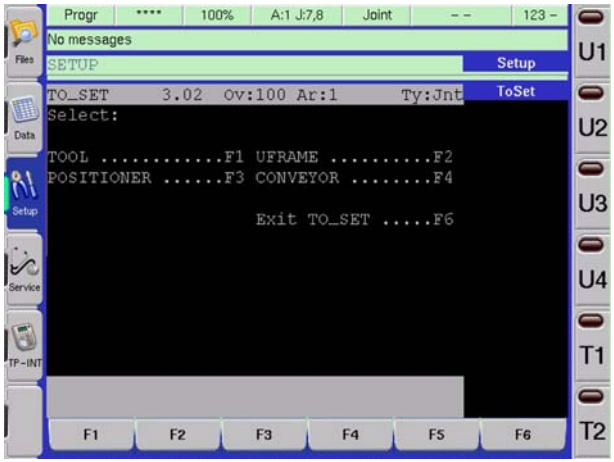
- **Generalità sui Programmi**
  5 - PDL2 Programs - Basic concepts
  6 - Programs structure overview
  7 - Life cycle of a PDL2 Program

1 TOOL and FRAME automatic calculation

To get the required precision in cartesian motions and to use a well defined frame of reference to teach working positions, it is needed to properly declare the TOOL dimensions and to define the wished User Frame (UFRAME).

These calculations can be executed by means of a program called TO_SET, available from the Teach Pendant, Setup page - ToolFrame subpage.

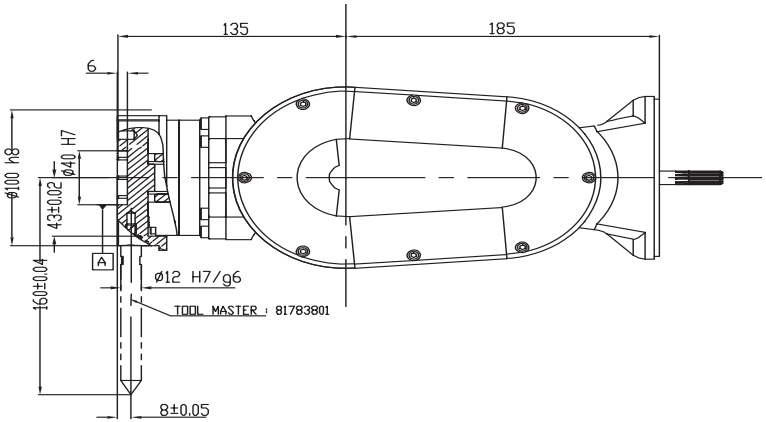
TO_SET home page is shown here beside.



Press **F1** key to access TOOL automatic calculation;

Press **F2** key to access UFRAME automatic calculation.

1.1 TOOL automatic calculation



In this environment it is possible to define the dimensions of the work tool mounted on the robot flange. The precision of the calculation is the same as the robot, used as a measuring instrument.

To execute the procedure, the following fixtures are required:

– **Calibrated tool (Tool Master)** - This is a calibrated tool with known X, Y, Z measurements that will be mounted on the robot flange for the acquisition of the reference position. As an alternative an identified point on the work tool can be used, for which the distance from the robot flange centre must be known and precise.
Let's take Smart NS 12-1.85 ARC robot as an example: the calibrated tool is:
dimension: 117 mm
In this case the calibrated tool must be mounted on the robot flange in x_tool direction, and the values to be declared areq: X=160, Y=0, Z= -8.

– **Reference point (or master cube)** - It is very important to identify a physical reference point in the environment surrounding the robot, so as to bring the robot TCP on to this point: once with the Tool Master then with the measuring tool. For this purpose a MASTER CUBE has been designed with a mobile tip that when fixed in zero position can be used as the REFERENCE POINT.
The cube (or any other point taken as reference) MUST BE IN A FIXED AND STABLE POSITION, easily accessible by the robot and positioned on the FLOOR (if the robot is on the floor), on the WALL (if the robot is on the wall), on the CEILING (if the robot is suspended from the ceiling).

After selecting the wished calculation, the program asks the user to insert the number of the TOOL to be either calculated or verified.

At this point the table that contains all the tools that already exist is scanned to check which of these conditions applies:

– The tool number entered corresponds to an empty position in the table, therefore the tool calculation proceeds with one of the available methods
– The tool entered already exists in the table , therefore the measurement and the method previously used for the calculation are displayed.

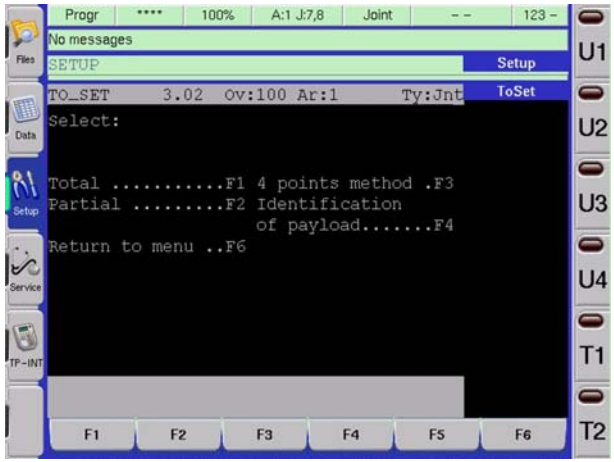
The following procedures are available:

– Tool calculation with standard method- Complete procedure
– Tool verification with standard method - Partial procedure
– Tool Calculation with "4 points method" - Complete procedure
– Local Tool verification with "4 points method".

1.1.1 Tool calculation with standard method- Complete procedure

It is required :
- **If it is the very first time that an automatic calculation of the tool is executed;**
- **If the reference point or the robot has been moved;**
- **If the robot has been recalibrated.**

The complete procedure can be used to store in UD: the scan result position and the declared tool master values in TO_SET.VAR . This procedure only has to be executed once. All the subsequent calculations of a tool mounted on the robot flange will only require the Partial procedure.



a. Press **F1** to execute the standard complete procedure (see figure here beside).

b. The tool master values are displayed; the user can either modify them (**F2**) or continue (**F1**).

c. Move the TOOL MASTER TCP in the refernece position, with the corresponding values properly definede.

d. When the robot is on the position, press F1 to get the position.

Go on with the Tool verification with standard method - Partial procedure steps.

1.1.2 Tool verification with standard method - Partial procedure

It is to be executed only if the Total Procedure has already been executed at least once.

With the partial procedure the measurements of the tool mounted on the robot flange can be calculated . It is only possible to calculate the relocations (X,Y,Z); or continue in the calculation of the rotations (<A>: Euler 1; <E>: Euler 2) and lastly it is possible to calculate the rotation around the new Z- tool (<R>: Euler 3).

a. Move the TCP of the TOOL TO BE MEASURED on the reference point. The robot may be in any position: the operator can position the TCP on the reference as preferred, according to requirements. Vedere figura a destra in alto.

b. When the robot is on the position, press F1 to get the position.

c. The calculated values of $TOOL are displayed. It is then possible to move axes 4,5,6 in TOOL to check that the new TCP does not move on the reference (movements of approx. 3 millimetres are accepted).

c.1 If the TCP moves, check the robot calibration, tool master dimensions declared in the Complete procedure, position of the reference point (not changed as to the Complete procedure).

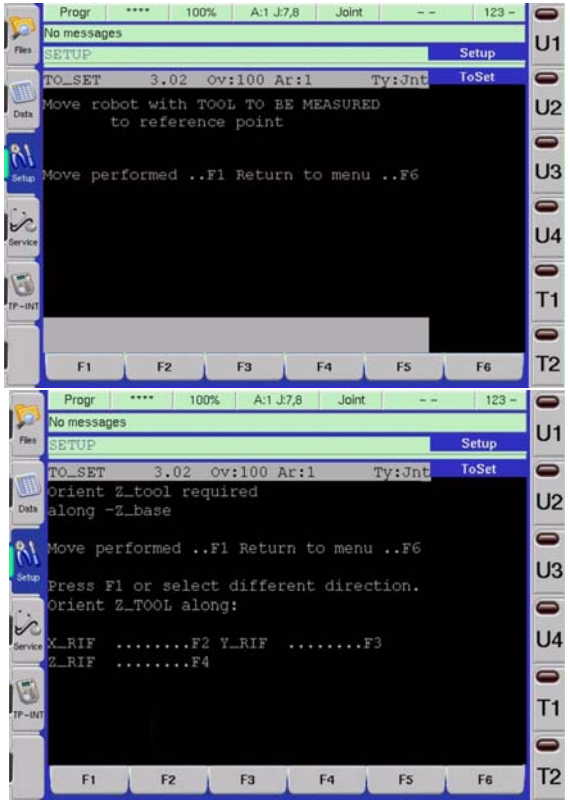
c.2 Press **F3** to repeat the procedure.

d. Press **F2** to save the calculated values.

e. Press **F1** to go ahead in the procedure and calculate rotations too.

f. Choose the direction along which orientate the tool Z-TOOL, parallel to one of the robot half-axes: **F2**..**F4**, keys, as shown in the low right figure.

g. Direct the Z-TOOL in relation to the chosen direction.



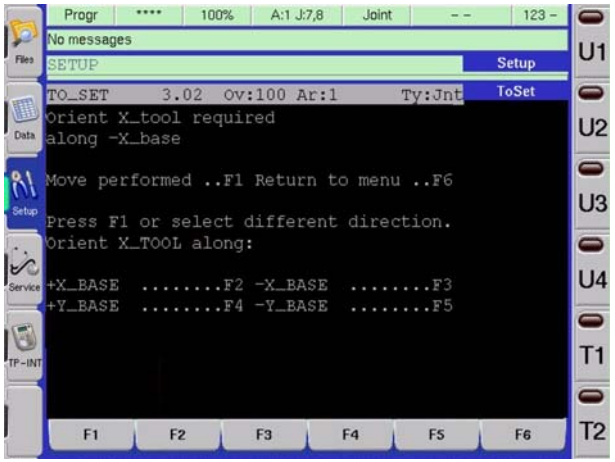
h. Press **F1** when the motion is finished.

i. Check if the new orientation is the wished one: if NOT, press **F3** to execute the procedure again.

j. Press **F2** to save the calculated data (relocations and orientations).

k. Press **F1** to continue.

l. Choose the direction along which orientate the tool X-TOOL, parallel to one of the robot half-axes (the axis chosen at step f. is no longer available): **F2**..**F5** keys, as shown in the right figure.



m. Direct X-TOOL in relation to the chosen direction.

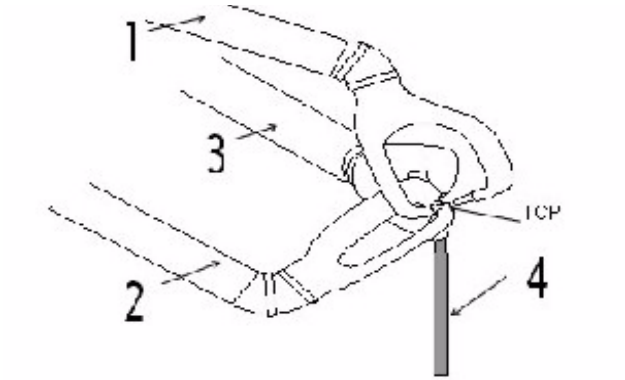
n. Press **F1** when the motion is finished.

o. Check if the new orientation is the wished one: if NOT, press **F3** to execute the procedure again.

p. Press **F2** to save the calculated values (relocations and orientations). After every Save operation, if there are already Euler angles in the table, the calculated values of $TOOL are displayed and the user is asked whether they have to be kept or to be overwritten with new data. This makes it possible to keep the Euler angles if it is only required to recalculate the tool relocations.

q. At the end of the procedure, if the user tries to exit without saving the calculated values, the program warns the user and allows to save data, if wished.

1.1.3 Tool Calculation with "4 points method" - Complete procedure



This procedure brings the TCP to a visible reference point within the robot working space, with at least three different orientations and acquires the relevant positions (see figure on the left).

```
1   acquisition point 3
2   acquisition point 2
3   acquisition point 1
4   reference
```

The procedure steps are described as follows:

a. Bring the TCP on the reference point. See the low left figure.

b. Press **F1** at the motion end, to get position 1.

c. Bring the TCP back to the reference point with a sufficiently different orientation as to the previous point.

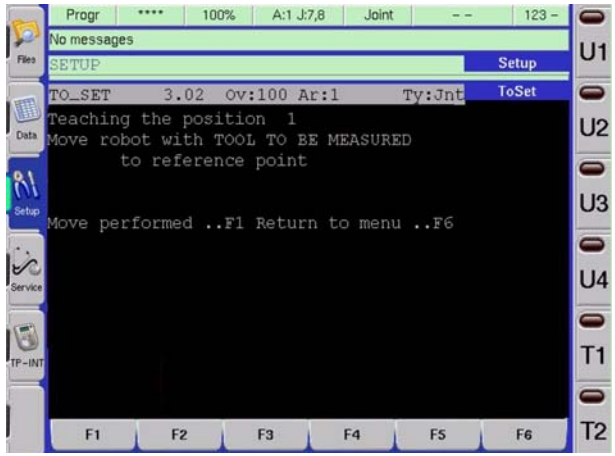
d. Press **F1** at the motion end, to get position 2.

e. Bring the TCP back to the reference point with a sufficiently different orientation as to the previous points.

f. Press **F1** at the motion end, to get position 3.

g. If the TCP measuring is possible, the user is provided with an evaluation of the acquisition procedure accuracy (good, imprecise, not reliable) together with the possibility of either accepting the values and going ahead with the orientation calculation (see step f. of the Tool verification with standard method - Partial procedure),

h. or going on with other positions acquisition (max 8 points), as described in the previous steps, moving the robot in Tool modality, if wished.



i. End of points acquisition : it is possible to continue with the orientation calculation, repeat the calculation of X, Y and Z, return to main menu.

If the user has acquired 8 points and the algorithm has not been able to calculate the tool, even with an imprecise measurement, this means that the acquired points are imprecise or are dependently linear
The more diversified the orientation of the acquired points, the better the results!

1.1.4 Local Tool verification with "4 points method"

It is to be executed only if the Total Procedure has already been executed at least once.

This procedure is used to repeat a tool measurement that has already been executed with the “4 points method”. It is very similar to the Tool Calculation with "4 points method" - Complete procedure, with the additional possibility to move the reference on the remote TCP with MOVE commands to the position stored together with the tool measurement. In this way the measurement repeatability is improved since the tool verification is executed acquiring the points with the same orientation (with a very slight approximation) as those of the previous acquisition.

a. With the Controller in DRIVE ON state, press **START** key to move the robot with the tool to be measured, towards the first stored position. The motion speed is the one displayed on the Teach Pendant screen.

b. Press **F1** at the end of the movement, to get the position.

c. Go ahead with the nex acquisition, as already described in the Tool Calculation with "4 points method" - Complete procedure.

d. Once the algorithm is able to calculate a tool, a screen page is displayed informing the user that as from that moment it is possible to move in TOOL reference.

1.2 UFRAME automatic calculation

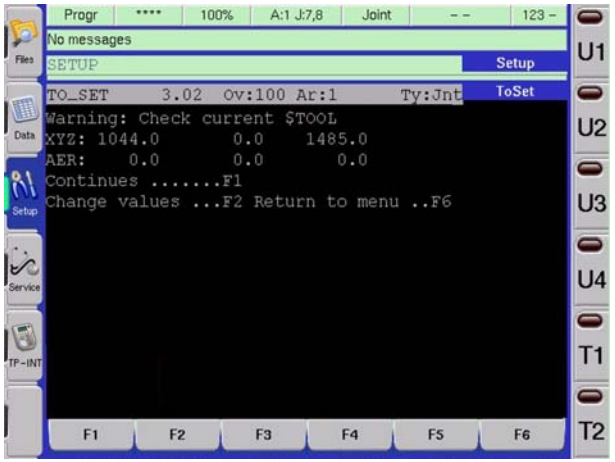
This environment is entered by pressing key F2 from the main menu of TO_SET. In this environment the new reference system can be defined in relation to which the work positions will be learnt. The precision of the calculation is the same as the robot, used as a measuring tool

To execute this procedure it is required to have:

– **Tool with known dimensions mounted on Robot Flange** - A reference tool must be mounted on the flange that has known dimensions declared in the $TOOL system variable.
It is advised to use the TOOL MASTER or CALIBRATED TOOL that was used for the Tool automatic calculation (see paragraph TOOL automatic calculation).
As an alternative an identified tip on the work tool can be used that has a distance from the robot flange centre that is known and precise.

– **3 Reference Points (ORIGIN, Xpos and XYpos)** - 3 reference points have to be identified on the workpiece :
  - ORIGIN: where the FRAME origin will be fixed;
  - Xpos: X-frame direction and sense;
  - Xypos: Y-frame direction and sense.

The steps of the procedure are described here below:



a. check the current $TOOL values (see the figure on the right).
The values are to refer to the tool mounted on the flange and that will be used for the FRAME calculation. The values are displayed and they can be modified if necessary. Press **F2** to modify them.

b. Press **F1** to continue the procedure.

c. Move the tool TCP to ORIGIN position. **The robot may be in any position: the operator can position the TCP on the ORIGIN point as preferred, according to requirements.**

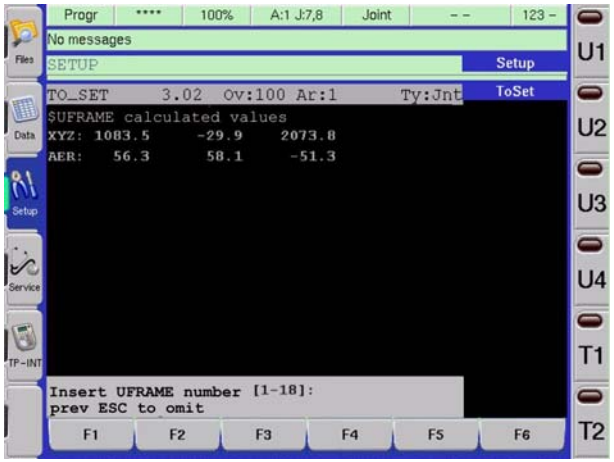
d. Press **F1** at the end of the movement, to get the position.

e. Move the tool TCP to Xpos position. **The robot may be in any position: the operator can position the TCP on the Xpos point as preferred, according to requirements.**

f. Press **F1** at the end of the movement, to get the position.

g. Move the tool TCP to XYpos position. **The robot may be in any position: the operator can position the TCP on the Xpos point as preferred, according to requirements.**

h. Press **F1** at the end of the movement, to get the position.



i. The program displays the $UFRAME calculated values (XYZ relocations and AER orientations). Now it is allowed to move in UFRAME modality (select the COORD field, from Motion Page) to check that the new origin is the desired one. If not, check the following:
   - Robot calibration;
   - $TOOL values (referred to the tool mounted on the flange);
   - the 3 reference points (check that the sequence required by TO_SET has been followed)

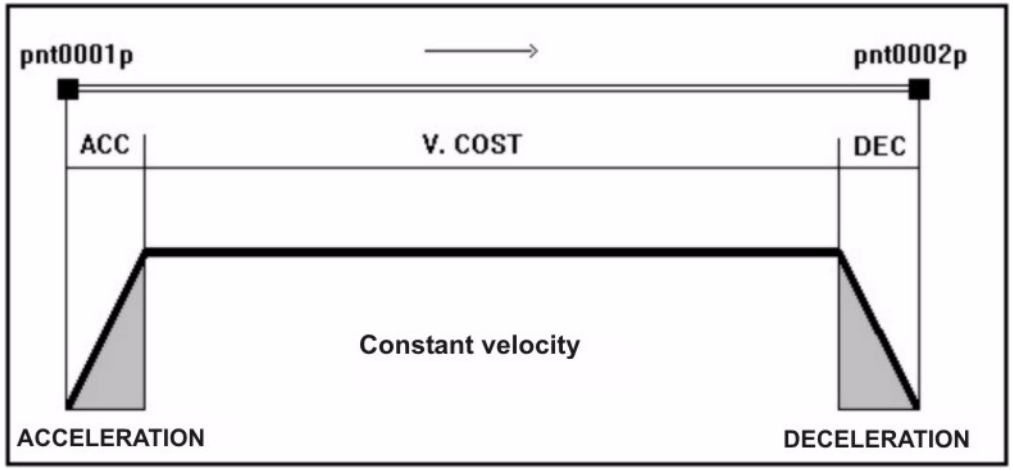
   make the modifications and press F1 to execute the procedure again, starting from step a.

j. SAVE the calculated values, in the TU_FRAME (press **F2**, insert the wished UFRAME number and press **ENTER** to confirm).

2 Types of movements overview

As already explained, the System takes into account the following reference frames:

World frame	–	Workshop reference frame in relation to where the machines are positioned
Base frame	–	frame that indicates the robot base
User frame	–	frame that indicates the workpiece
Flange frame	–	frame that indicates the robot flange
TCP frame	–	frame that indicates the tool tip



A movement starts with the arm **acceleration** along a specified trajectory, continues with a **constant velocity motion** and ends with a **deceleration**, until the final position is reached (see the figure here beside: motion from **pnt0001p** to **ont0002p**).

A movement is distinguished by:

– Trajectory
– Speed Control
– Acceleration and Deceleration
– Motion termination
– Continuous Motion

2.1 Trajectory

It represents an Arm motion from an initial position to a final position. The trajectories can be classified as follows:

– joint trajectory: JOINT

– linear trajectory: LINEAR

– circular trajectory: CIRCULAR.

JOINT Interpolation

During joint interpolation ($MOVE_TYPE := JOINT or MOVE JOINT TO), the joint angles of the arm are linearly interpolated from their initial to final values. All axes start moving at the same time and reach their destination at the same time. The path followed by the tool centre point (TCP) is not predictable, although it is repeatable.

Joint interpolated movements between two positions are always possible.

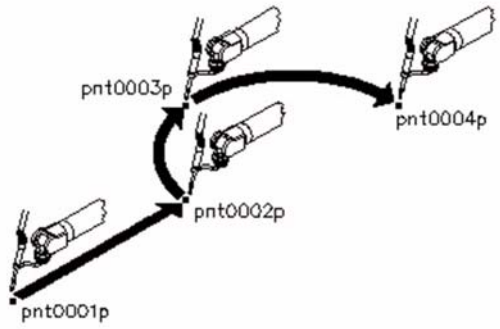
LINEAR Interpolation

During linear interpolation ($MOVE_TYPE := LINEAR or MOVE LINEAR TO), the TCP moves in a straight line from the initial position to the final position. The orientation of the tool also changes from the initial position to the final position according to the mode defined by the $ORNT_TYPE variable. This specific program variable can have the values of the following predefined constants: RS_WORLD, RS_TRAJ, EUL_WORLD, WRIST_JNT.

CIRCULAR Interpolation

During circular interpolation ($MOVE_TYPE := CIRCULAR or MOVE CIRCULAR TO), the TCP follows a circular arc from the initial position to the destination. An additional position, called the VIA position, must be specified to define the arc. Only the location component of the VIA position is used; its orientation does not affect the motion.

As for linear interpolation, the $ORNT_TYPE predefined variable indicates the type of evolution of attitude must be performed.



2.1.1 Orientation Evolution during Linear or Circular movements

The orientation of the tool during linear and circular movements evolves from the initial position to the final position in the manner indicated by the $ORNT_TYPE variable. Possible values of this specific variable of the program are as follows:

– RS_WORLD (two-angles related to the world frame)
Orientation interpolation is done by linearly interpolating the values of two rotation angles, tool rotation and tool spin. The tool rotation angle is the angle about the common normal between the beginning tool approach vector and the destination approach vector. The tool spin angle is the angle about the approach vector from the beginning position to the destination position. The evolution is related to the World frame independently from the trajectory. RS_WORLD is the default value for $ORNT_TYPE.

– RS_TRAJ (two angles related to the trajectory)
Orientation interpolation is done in the same way than RS_WORLD but the rotation and spin angles are related to the trajectory. This is particularly useful during circular trajectory having an angle grater than 180 degrees when the user wants to maintain the tool orientation constant with respect to the trajectory. During linear motions the orientation evolution is the same than RS_WORLD.

– EUL_WORLD (three-angle)
Orientation interpolation is done by linearly interpolating the values of the three Euler angles of rotation, E1, E2, and E3.

– WRIST_JNT (wrist-joint)
Orientation interpolation is done by using a combination of joint and linear interpolation. This permits the tool to move along a straight line while the wrist joints are interpolated in joint coordinates. The starting and ending orientation will be used as taught, but because of the joint interpolation, the orientation during the movement is not predictable, although it is repeatable. For example, using either EUL_WORLD or RS_WORLD, if the beginning and ending orientations are the same, then the orientation of the tool will remain fixed during the motion. However, with WRIS_JNT orientation interpolation this is not guaranteed. However, WRIST_JNT orientation control allows for much smoother motion near wrist singularities.

3 Motion Dynamics

3.1 Speed Control

Predefined variables are used to control the maximum or constant speed of a motion, some of which are system-wide, some of which are program-specific, and some of which are arm-specific. Two kinds of speed controls are used:

– absolute speed values or limit values, measured in normal speed units such as radians per second or meters per second;
– override values that govern the absolute speed values, measured in percentage.

3.1.1 Speed Overrides

The speed overrides (percentage speed values) that apply to all motions are as follows:

– $GEN_OVR allows the operator to change simultaneously the acceleration, speed and deceleration values of Motion programs. Since this influences the acceleration, speed and deceleration values in a coordinated manner, the trajectories are usually maintained (unless there are servo tracking errors) when this variable is changed. It is common to the whole system and can be changed from the Teach Pendant. The PDL2 programs can read it only.
– $ARM_OVR permits acceleration, speed, and deceleration of a specific arm to be modified from within a program. Since it affects acceleration, speed, and deceleration in a coordinated way, trajectory shapes are generally maintained when this control is modified (except for differences in servo following errors at different values of override).
  Note that if constant speed during transitions between continuous motions is more important than constant trajectory, then $ARM_SPD_OVR or $PROG_SPD_OVR should be used (acceleration and deceleration will be left at current values as these speed overrides are reduced).
– $ARM_SPD_OVR permits the speed of a specific arm to be modified by a program. Acceleration and deceleration are not affected. This implies that the shape of continuous motion trajectories may change as this control is changed.
  There is one $ARM_DATA structure per arm, and the ARM_SPD_OVR field has a default value of 100%.
– $PROG_SPD_OVR permits speed of all motions issued from a specific program to be modified by the program. Acceleration and deceleration are not affected.
  It is a program-specific variable with a default value of 100%.

$GEN_OVR and $ARM_OVR take effect during motions. That is, if either variable is changed while a motion is in progress, the current motion will speed up or slow down accordingly. However, a change in $PROG_SPD_OVR or $ARM_SPD_OVR will not take effect until the next move statement within the program for which it is defined.

The required percentage value can be assigned either with MODAL method or with NODAL method.

The MODAL assignment is **permanent** (until the next modal assignment), whereas the NODAL assignment is **temporary** (just for the MOVE statement it is associated to).

Example:

```
$ARM_SPD_OVR := 50 -- modal assignment: current speed 50%

MOVE LINEAR TO pnt0001p,
  WITH $ARM_SPD_OVR=10 -- nodal assignment: temporary speed 10%
ENDMOVE
-- current speed 50%
```

3.1.2 Cartesian Speed Control

Under normal operation, the following two predefined variables govern the basic speed of a Cartesian motion (linear or circular), with any type of orientation control.

– $LIN_SPD_LIM determines the linear speed. It is a variable for each Arm, whose value depends on the Robot and cannot be modified by the User.
– $ROT_SPD_LIM determines the rotational speed. It is a variable for each Arm, whose value depends on the Robot and cannot be modified by the User.

The component of motion that takes the longest to get from its initial to final location will move at the programmed speed limit, reduced by the total override. All of the other components will move at lower than programmed limits, so that all components start and stop at the same time.

3.1.2.1 Cartesian Speed Control Options

The process of determining which component governs Cartesian speed is called preplanning. This occurs just before the motion actually occurs (i.e. after each MOVE). It is possible to force the preplanner to pick a particular component of motion using the predefined variable $SPD_OPT. It is a program-specific variable (each program can have its own value) that can be assigned the following predefined constants:

– SPD_CONST is the default value. It moves the arm at constant speed with the SMS being picked by the preplanner, as explained previously.
– SPD_JNT moves the arm along the requested Cartesian trajectory, at the maximum speed of at least one joint. The TCP will not move at constant speed.
– SPD_LIN moves the TCP according to the requested $LIN_SPD, forcing the rotation component to move in coordinate way.

3.1.3 Joint Speed Control

The speed override for each joint is determined by $JNT_OVR, which permits acceleration, speed, and deceleration to be modified together by a program. There is an override value for each axis of each arm. The default values are 100%.

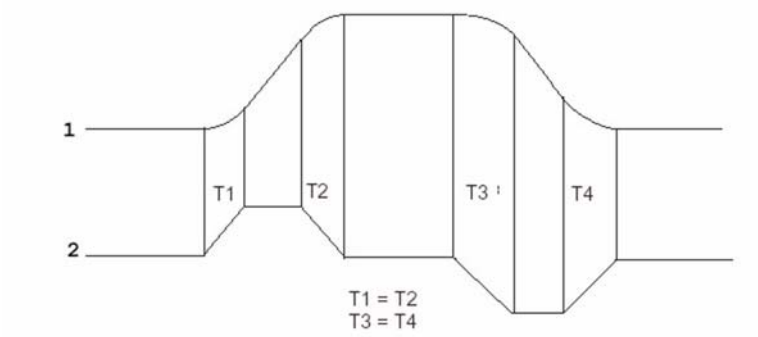
3.2 Acceleration and Deceleration

Acceleration and deceleration in the C4G controller are each broken into three phases, a constant jerk phase, followed by a constant acceleration phase, followed by another constant jerk phase.

Currently, C4G forces the acceleration profile to be symmetric and the deceleration profile to be symmetric.
This means that the constant jerk phases during acceleration (T1 and T2) are the same length and the constant jerk phases during deceleration (T3 and T4) are the same length.

As with speed, acceleration and deceleration also have overrides. $GEN_OVR and $ARM_OVR not only affect speed, but also acceleration and deceleration.

In addition, arm specific and program specific variables exist to further override acceleration and deceleration. Such variables are arm or program specific:

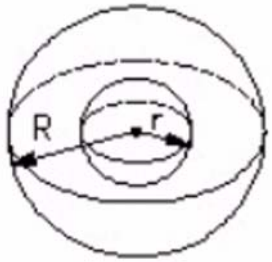


1 - speed
2 - acceleration

- $PROG_ACC_OVR permits acceleration of all motions issued from a specific program to be modified by the program. It is a program specific INTEGER with a default value of 100%;
- $ARM_ACC_OVR permits acceleration of a specific arm to be modified by a program. It is an INTEGER field of the array, $ARM_DATA, with one field for each arm. The default value is 100%;
- $ARM_ACC_OVR permits acceleration of a specific arm to be modified by a program. It is an INTEGER field of the array, $ARM_DATA, with one field for each arm. The default value is 100%;
- $ARM_DEC_OVR permits deceleration of a specific arm to be modified by a program. It is an INTEGER field of the array, $ARM_DATA, with one field for each arm. The default value is 100%.

Changes in these four overrides take effect only on succeeding motions, not during any current motion. When HOLD, CANCEL, LOCK, or DEACTIVATE are used, maximum deceleration is used regardless of the above override settings.

3.3 Motion termination

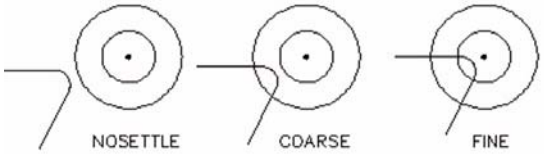


It specifies the accuracy in robot positioning on the motion final destination, for a NON FLY movement, before the next operation is performed.

For non-continuous motions, the predefined variable $TERM_TYPE is used to determine when the motion will be terminated based on how close the arm must be to its destination.

The predefined constants COARSE, FINE, JNT_COARSE, JNT_FINE, or NOSETTLE can be assigned to $TERM_TYPE, with NOSETTLE being the default. Syntax:

```
$TERM_TYPE := <predefined_constant>
```



- COARSE - must be used in Cartesian movements and indicates that the movement is accomplished if the TCP stays inside the sphere which is centered in the final destination, with a radius specified by $TOL_COARSE predefined variable, for a time greater equal to $TOL_ABT (anti-bounce time).
- FINE - must be used in Cartesian movements and indicates that the movements is accomplished if the TCP stays inside the sphere which is centered in the final destination, with a radius specified by $TOL_FINE predefined variable, for a time greater equal to $TOL_ABT (anti-bounce time).
- NOSETTLE - No controls are executed; NOSETTLE is the default setting. It represents the lowest accuracy level.
- JNT_COARSE - must be used in Joint movements and indicates that the movement is accomplished if EACH JOINT is less far than a distance (in degrees) specified by $TOL_JNT_COARSE associated to the corresponding axis.
- JNT_FINE - must be used in Joint movements and indicates that the movement is accomplished if EACH JOINT is less far than a distance (in degrees) specified by $TOL_JNT_FINE associated to the corresponding axis .

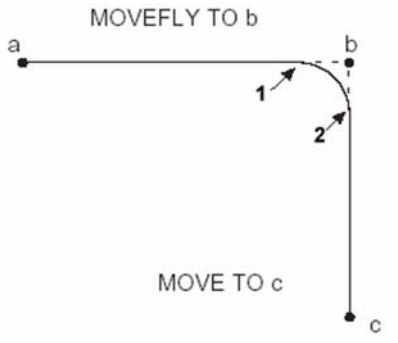
The predefined variables $TOL_COARSE and $TOL_FINE indicate the tolerance values. The default values for $TOL_COARSE and $TOL_FINE are arm dependent.

The predefined variable $TOL_ABT (anti-rebound time) indicates the time during which the arm is to be within the specified tolerance before the motion is declared completed/finished.

The predefined variables $TOL_JNT_COARSE and $TOL_JNT_FINE indicate the COARSE and FINE tolerance values respectively, for each joint, measured in degrees.

Note that tolerance settings are joint settings, not tool centre point settings. To obtain the correct tool centre tolerance values if the tool itself is large, it is necessary to use lower tolerances than those used for smaller tools; in this case make reference to the Cartesian frame thresholds (Fine, Coarse)

3.4 Continuous Motion



1. Movement start towards c
2. Movement end towards b

The Continuous Motion allows the Program execution without stopping the Arm on the taught positions.

To indicate a countinuous motion, the MOVEFLY statement is to be used, instead of the MOVE statement. If one more movement follows the MOVEFLY, the Arm will not stop on the first destination, moving instead from the starting position to the final position of the second movement, without stopping on the common position of the two movements.

Fly movements between Joint trajectories an Cartesian trajectories are not allowed (NO n y).

The predefined variable $FLY_TYPE determines which of several different algorithms are used to control continuous motion. The algorithm used affects not only the speed and shape of the trajectory during fly motions, but also the amount of stress placed on the arm and on the component or tool connected to it. The available $FLY_TYPE values are FLY_NORM and FLY_CART.

FLY_NORM

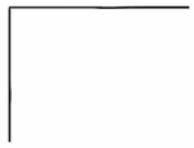
With $FLY_TYPE set to FLY_NORM, MOVEFLY causes the deceleration of the issued motion to be overlapped with the acceleration of the next motion.

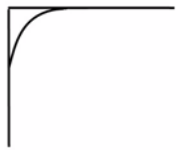
The predefined variable, $FLY_PER (the only one associated to FLY_NORM), can be used to reduce the time in fly and to bring the trajectory closer to the intermediate taught position. FLY_NORM is the default value.

$FLY_PER represents the percentage of the delay in executing the fly motion. Decreasing its value, causes the delay of the fly motion to increase and the trajectory to be closer to the MOVEFLY destination (see the here below figure).

For example, if the value of $FLY_PER is 100%, the fly begins at the start of deceleration of the fly motion. If $FLY_PER is 75%, then fly will begin after 25% of the deceleration is finished (75% will be combined with the next motion).

```
ROUTINE call_prog_1
BEGIN
  $FLY_TYPE := FLY_NORM
  $FLY_PER := 50
  MOVEFLY LINEAR TO pnt0001p ADVANCE
  $FLY_TYPE := FLY_HIGH
END call_prog_1
```


$FLY_PER := 1


$FLY_PER := 30


$FLY_PER := 100

4 Introduction to the IDE programming environment

Programs used on the C4G Controller are written in PDL2 programming language. There are several methods to create such programs, depending on the programmer and the required application.

The main development environment is:

– IDE environment - this is a development environment used to develop motion programs on the Teach Pendant.

The IDE environment is used to edit a program that already exists, that is Holdable and has EZ attribute.

If the required program does not exist yet, use the **Prog** (**F1**) - **Create** (item **2**) command, from Prog Page on the Teach Pendant.

Prog (**F1**) - **Create** command

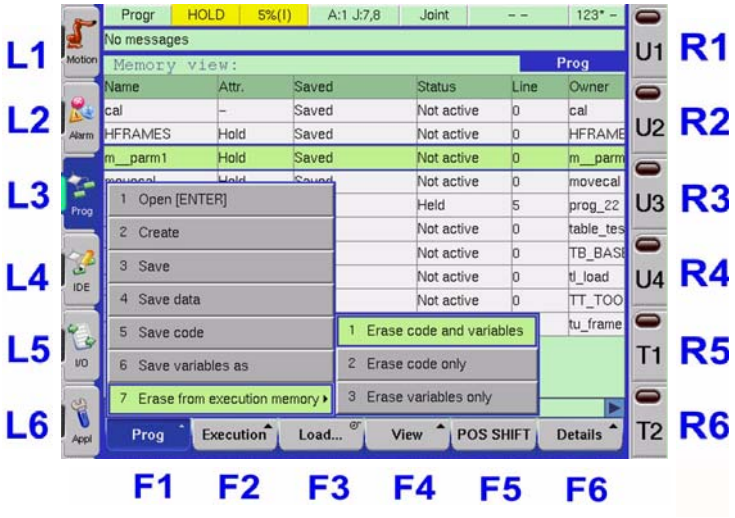
Allows to create a new program to be edited in IDE environment. The following situations might occur:

– the requested program already exists in execution memory

– the requested program only exists as a file in UD:. The command loads it in the execution memory (ML).

– the requested program does not exist. The command creates a .COD file and loads it in the memory.

To open in IDE the newly created program, either use **Prog** (**F1**) - **Open [ENTER]** (item **1**) command, or select it and press ENTER.



4.1 Programming in IDE (on TP) for a motion program

The required steps to develop a Program in IDE environment, are as follows:

a. Program Editing

b. Teaching positions (inserting a motion statement)

c. Program execution test

d. Saving the program and closing IDE

4.1.1 Program Editing

Let us now look at how to proceed to execute the main program editing functions: they are listed below.

– Selecting one or more text lines, deleting/undeleting, copying
– Move the Edit cursor
– Inserting a new instruction
– Commenting/Uncommenting a program line
– Changing an existing instruction
– Inserting/viewing /deleting variables
– Importing from other programs
– Viewing a program in pages.

4.1.1.1 Selecting one or more text lines, deleting/undeleting, copying

a. press the Selection function key (F4) from the main menu of the IDE Page (see fig. on the left)

b. in the pop-up menu that is opened by the system, choose the Enable Marking function and move in the editing area with the cursor keys, to execute marking; there are 3 functions in the central menu (see fig. on the right)

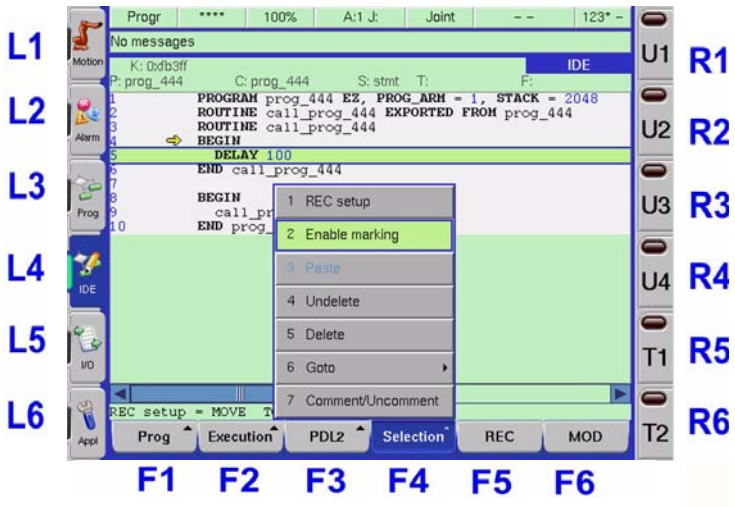
b.1 to cut the selected part of the text, use the Cut - cuts the selected lines command (F1)

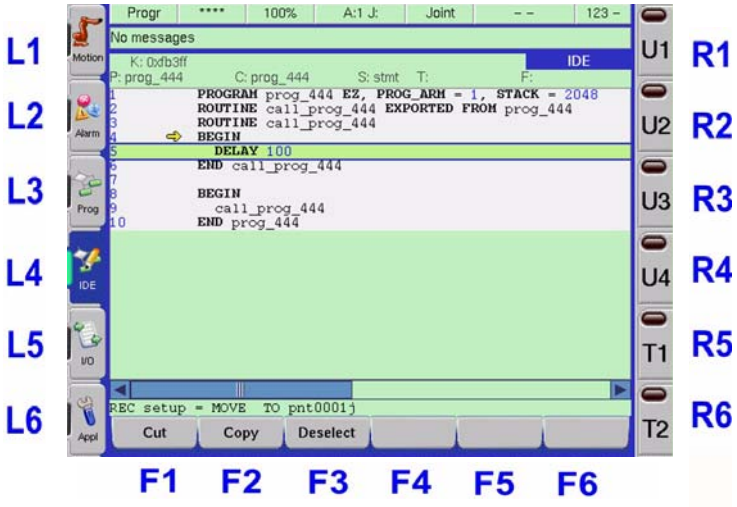
b.2 to copy the selected part of the text, use the Copy - copies the selected lines command (F2)

b.3 to cancel the marking, choose Deselect - deactivates the marking (F3)

c. to insert (paste) the previously cut or copied part of the text, use the Paste function in the pop-up menu. See. item 3 in the fig.on the left.

d. to undelete the previously cut text (to cancel deletion), use the Undelete (item 4) function in the pop-up menu; the text is inserted underneath the current position of the EDIT CURSOR.





Several program lines can be selected using the SHIFT+cursor combination.

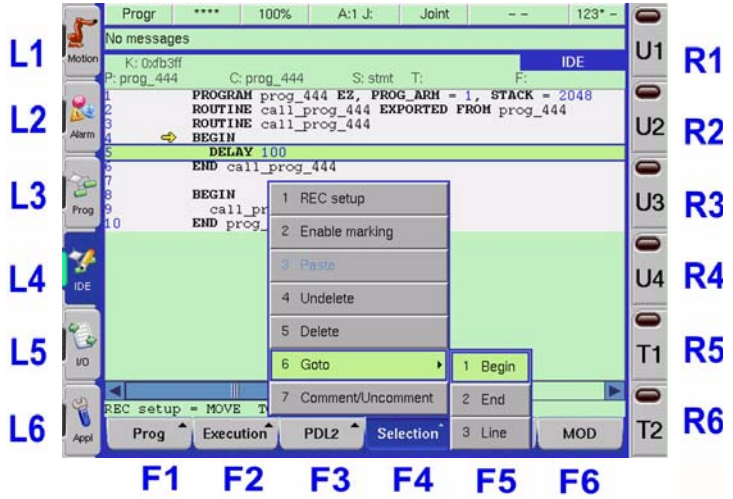
4.1.1.2 Move the Edit cursor

To move the EDIT CURSOR, the easiest way is to use the cursor keys; if it is wished to move it to a specific position, proceed as follows:

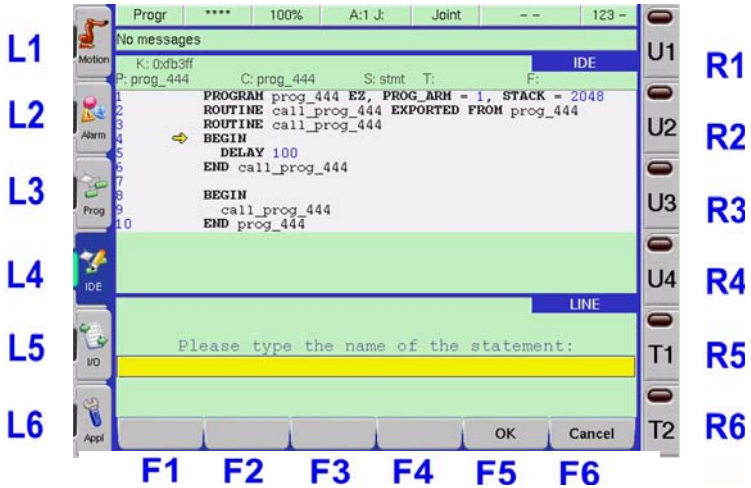
a. press the Selection function key (F4) from the main menu of the IDE Page;

b. in the pop-up menu that is opened by the system, choose the Goto function (see fig. here beside)

b.1 choose either the BEGIN instruction, or the END instruction or indicate a precise line number.



If it has been decided to indicate a line number, the system opens a dialogue window; check that the Alphanumeric keypad is in '123' mode and insert the required number, confirming with 'OK' (F5) or ENTER (see fig.below).



4.1.1.3 Inserting a new instruction

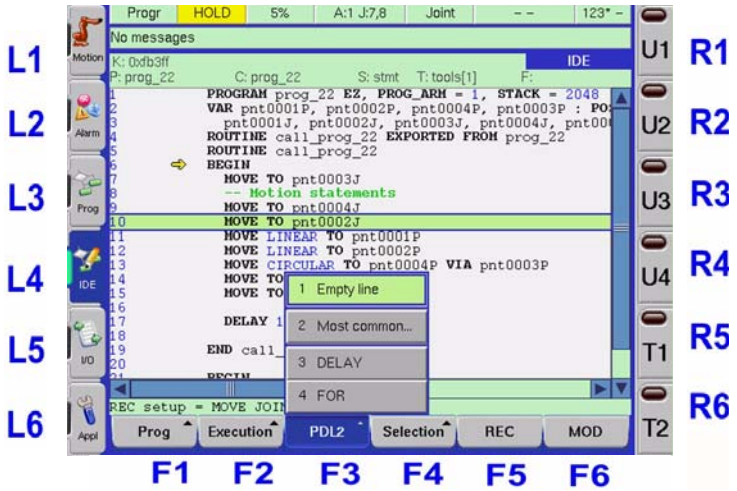
a. press the PDL2 function key (F3) from the main menu of the IDE Page;

b. the system shows the last 9 instructions used, plus the “Empty Line” option

b.1 if the required instruction is in the list, select it and confirm with ENTER: it will be inserted under the line currently pointed by the EDIT CURSOR.

b.2 if it is not in the list, choose 'Empty Line' to open a new program line under that where the EDIT CURSOR is pointing. Insert the new instruction using one of these two methods:

b.2.1 guided insertion, with the aid of a (Template): press the Statement function key (F1)., choose the instructions category, press ENTER and choose the required instruction. Fill the template fields in;

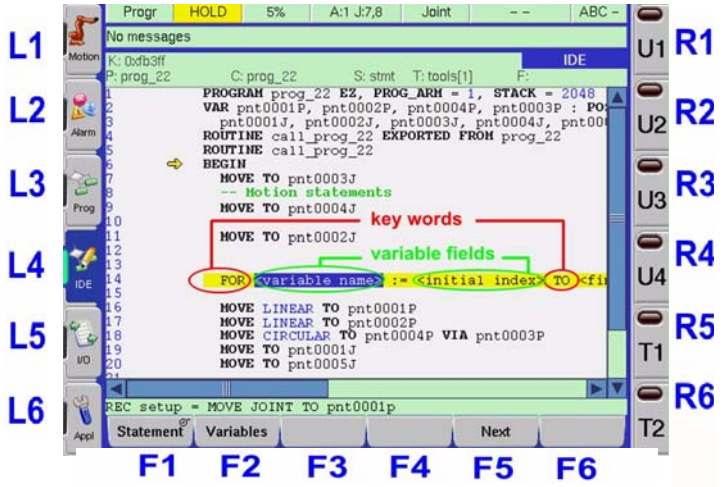


b.2.2 direct insertion from the Alphanumeric keypad.

c. Regardless of the chosen method, confirm with ENTER. The system will check the inserted program line and indicate any errors.

Template: A template is a sort of “mask” to help the user when inserting and/or changing an instruction. It is composed of key words (inserted by the system) and variable fields (that have to be completed by the user). The key words are written in bold high case characters on a light blue background. A variable field is confined by the characters '<' and '>'; when on it, the background becomes blue.

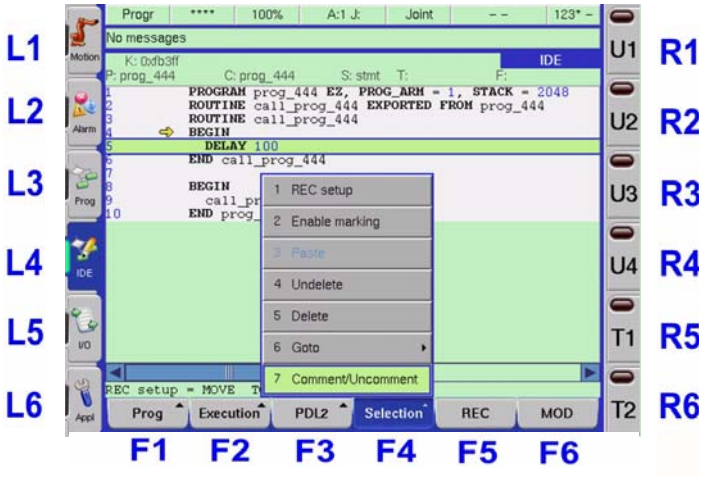
To move among the different variable fields it is highly recommended to use the Next function. See **F5** softkey, shown in the here beside figure.
Moving among the variable fields with the Next, key, IDE automatically updates the use mode of the Alphanumeric keypad (seventh field of the Status bar).



4.1.1.4 Commenting/Uncommenting a program line

a. press the Selection function key (F4);

b. in the pop-up menu that is opened by the system, choose the Comment/Uncomment function. This softkey acts as a toggle key: each time it is pressed it reverses the current state. (see the here beside figure)

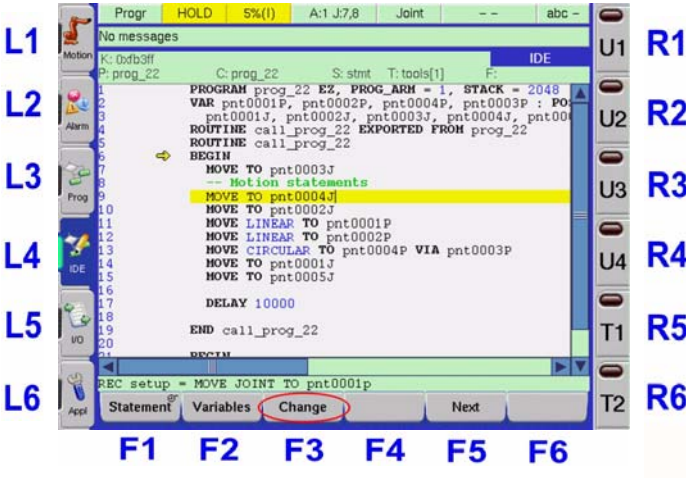


4.1.1.5 Changing an existing instruction

a. Position the EDIT CURSOR on the line involved and press ENTER: this gives access to the editing mode;

b. use the Alphanumeric keypad pad or the commands of the Bottom Menu, in the same way as described to complete the Statement command.

Special case: Changing a MOVE instruction. A particular case is to change a MOVE instruction. In this case there is the Change function, associated to key F3, which can be used to change the MOVE parameters. (see the here beside figure).



4.1.1.6 Inserting/viewing /deleting variables

To insert a new variable in the line of the program currently pointed by the EDIT CURSOR, proceed as follows:

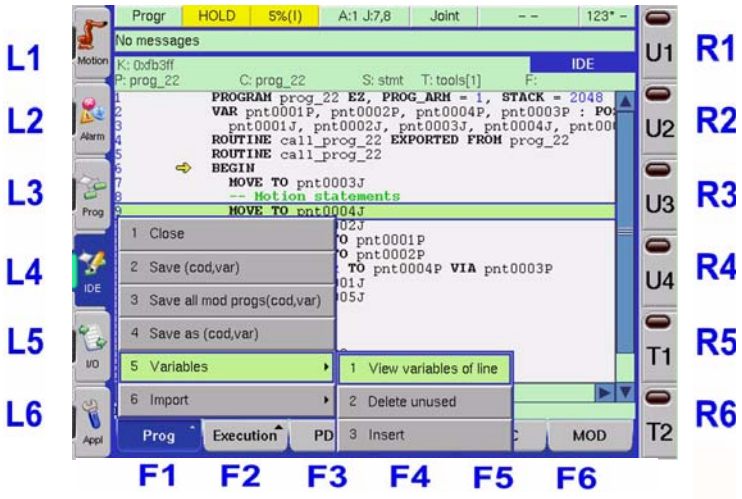
a. press the Prog softkey (F1) from the main menu of the IDE Page

b. Select the Variables function

c. choose item 3 - Insert

d. The system opens a dialogue window for the user to enter the name and type of variable required, choosing from the list displayed, or typing it in directly from the Alphanumeric keypad.

e. after making the changes, confirm with ENTER.



Viewing the variables used in a line of the program:

a. position the EDIT CURSOR on the required program line

b. press the Prog softkey (F1) from the main menu of the IDE Page

c. select the Variables, function from the main menu of the IDE Page

d. choose item **1** - **View variables of line**.

It is also possible to delete the declaration of the unused variables:

a. press the Prog softkey (F1) from the main menu of the IDE Page

b. select the Variables function from the main menu of the IDE Page

c. choose item **2** - **Delete unused**

4.1.1.7 *Importing from other programs*

To import declarations with the EXPORTED FROM clause from another program:

a. press the Prog softkey (F1) from the main menu of the IDE Page,

b. select the Import function,

c. according to the item to be imported, one or two dialogue windows are displayed. To import Variables and Routines there are two dialogue windows; for Types there is only one. The user has to answer the questions then press ENTER to confirm.

The classes of importable declarations area:

– Routine

– Type

– Variable.

4.1.1.8 *Viewing a program in pages*

To view a program page by page, use the Page Up and Page Down keys.

4.1.2 Teaching positions (inserting a motion statement)

In the IDE environment positional variables can be assigned the current arm position, inserting also the associated move instructions and corresponding declarations of these positional variables in the program code.

The basic needs about teaching positions are as follows:

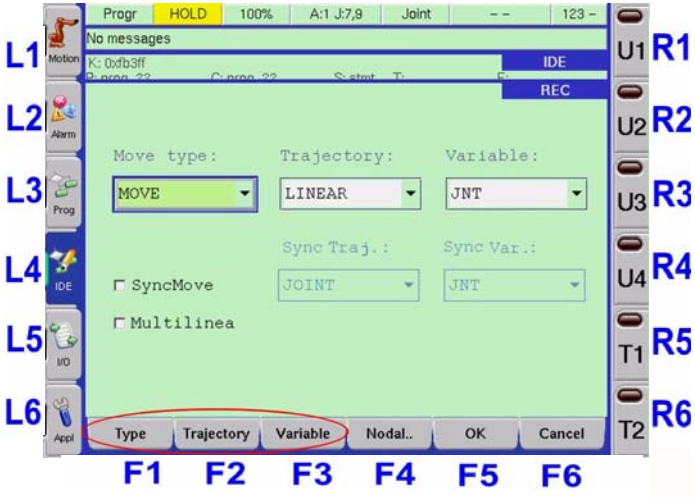
– Setup the REC key
– Teaching new positions
– Changing existing positions.

There is also the possibility of Nodal Setting (OPTIONAL).

In case of WiTP, the REC hardkey DOES NOT exist: its functionality, in IDE environment, is performed by the REC (F5) softkey , anyway available on TP4i too.
In the IDE screen there is the low status window, containing the current settings of the REC key (F5 or hardkey for TP4i).

4.1.2.1 Setup the REC key

To set the move instruction characteristics that are inserted in the program, each time the REC key is pressed.



At the opening of IDE, the REC key is set in joints modal mode, if it has not been set previously. To change this setting, either use the current REC Setup function, or activate the function directly by simultaneously pressing SHIFT+REC.

a. press the Selection function key (F4) from the main menu of the IDE Page

b. in the pop-up menu that is opened by the system, choose the REC setup function and follow the instructions of the system. The user can choose
   - the type of position variable - POSITION, JOINTPOS, XTNDPOS
   - the trajectory - DEFAULT, JOINT, LINEAR, CIRCULAR
   - fly or non fly.

It is to be noted that the new settings of the REC key are NOT maintained after a system restart. To make it permanent the configurations have to be saved (using the system command 'ConfigureSaveCategoryController /Shared').

4.1.2.2 Teaching new positions

After the REC key has been setup, proceed as follows:

a. position the EDIT CURSOR on the program line under which the new MOVE instruction is to be inserted

b. move the robot with the Jog keys to the position to be recorded

c. press the REC key. This operation inserts a MOVE instruction and the declaration of the positional variable used by the MOVE into the program.

4.1.2.3 Changing existing positions

If the contents of an existing positional variable are to be changed, proceed as follows:

a. move the EDIT CURSOR onto the program line to be executed

b. move the robot to the new position, by means of the Jog keys

c. press the MOD function key (F6) from the main menu of the IDE Page - the system shows a dialogue window in which a list of line positional variables is displayed

d. the user has to select the variable involved and confirm the change with ENTER or cancel it with ESC.

If, for the current MOVE, tool and frame settings are different than those currently set for the REC setup, the system displays an alarm message. It is up to the user to make the suitable changes.

4.1.3 Program execution test

As already mentioned, IDE is an integrated program editing and development environment. It is most important that all the programs are carefully checked before being run in automatic mode.

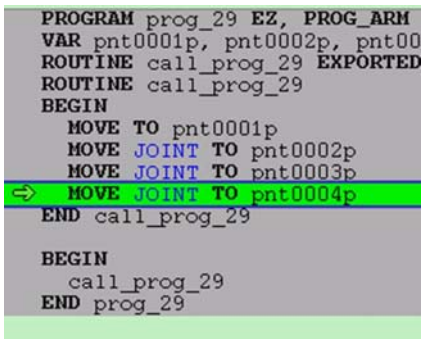
This paragraph describes some of the commands that influence the execution of the program currently opened in IDE, so as to be able to debug, identify any errors and make improvements.

- Activating/deactivating a program
- Aborting the execution of an instruction
- Setting step mode
- Move the execution cursor
- Inserting/removing a break point
- Bypassing an instruction
- Execute a temporary instruction

Displaying the currently being executed instruction.

In IDE environment, with Step Disabled, the execution of the program that is currently open can be observed in the Editor area.

If the opened program is running, IDE displays the instructions which are currently being executed, by focusing them with the EXECUTION CURSOR. The execution may be followed by the user by looking at the cursors moving through the program instructions.
See the example shown here beside (the execution cursor with dark green background indicates the currently being executed statement).

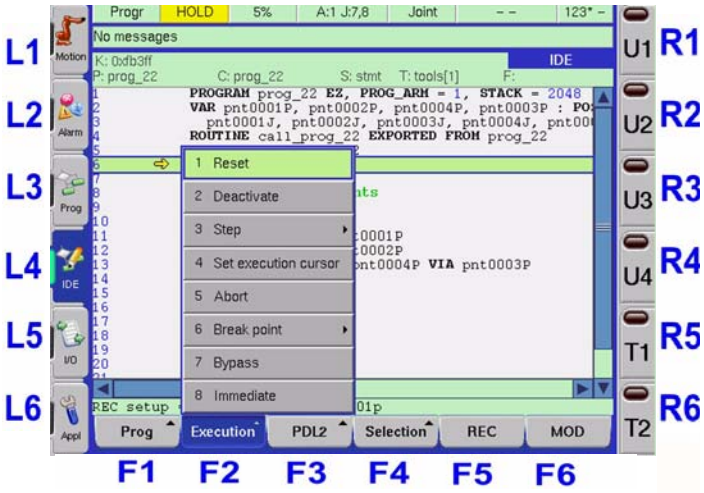


4.1.3.1 *Activating/deactivating a program*

The program currently open in IDE is always the ACTIVE one. To restart it after making changes, proceed as follows:

a. press the Execution function key (F2) from the main menu of the IDE Page;

b. in the pop-up menu that is opened by the system, choose the Reset function (item 1). - In this way the program is deactivated, then reactivated.



To deactivate the program only, proceed as follows:

a. press the Execution function key (F2) from the main menu of the IDE Page

b. in the pop-up menu that is opened by the system, choose the Deactivate function (item 2).

4.1.3.2 *Aborting the execution of an instruction*

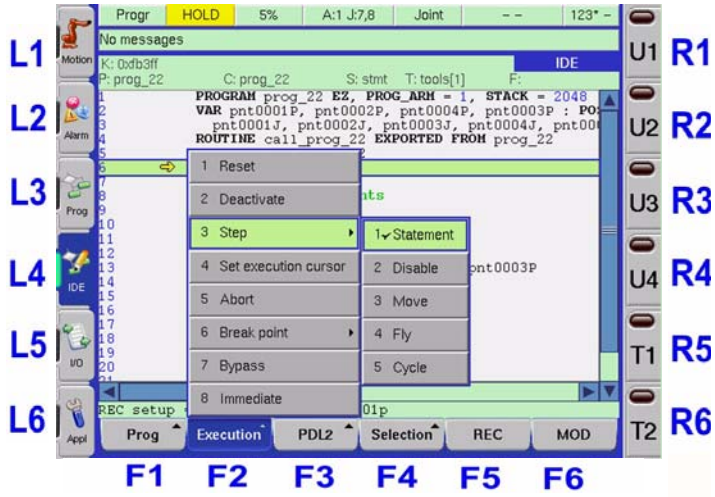
To abort the execution of the current instruction:

a. press the Execution function key (F2) from the main menu of the IDE Page (see the above figure);

b. in the pop-up menu that is opened by the system, choose the Abort function (item 5).

To continue to run the program from the instruction following that which has been aborted, press START.

4.1.3.3 Setting step mode



Usually the program execution is continuous, i.e. without interruption. The step execution serves to define the moment when the execution pauses, a new step is started each time the START key is pressed.

To set the step mode, proceed as follows:

a. press the Execution function key (F2) from the main menu of the IDE Page

b. in the pop-up menu that is opened by the system, move to the Step function (item 3)

c. from the new list opened by the system, choose the mode. The following modes are possible:
   - Statement (Single step) - The execution stops after each instruction. The protected Routines (the content of which is not displayed) are executed as a single statement; the execution of non-protected Routines (where the content is visible) stops at the end of each instruction.
   - Disable - Stops the program execution in steps.
   - Move - The execution stops after each single move. It cannot be executed on non-holdable programs.
   - Fly - To execute two or more motions in fly before suspending the program execution. It is similar to the Move, but the execution does not stop after the MOVEFLY instruction. It cannot be executed on non-holdable programs.
   - Cycle - Defines a single program cycle, that has to include the CYCLE or BEGIN CYCLE instruction, as a step.

4.1.3.4 Move the execution cursor

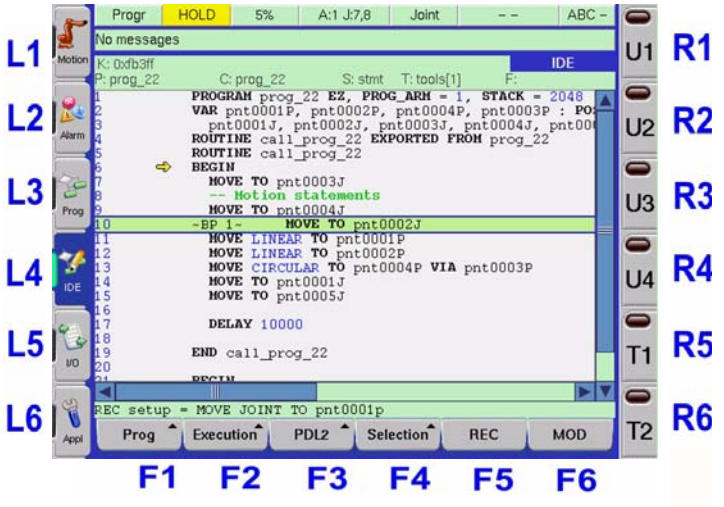
Moving the execution of the program to a line that is not that pointed by the EXECUTION CURSOR:

a. move the EDIT CURSOR onto the program line to be executed

b. press the Execution function key (F2) from the main menu of the IDE Page

c. in the pop-up menu that is opened by the system, choose the Set execution cursor function (item 4).

4.1.3.5 Inserting/removing a break point



A break point is an interruption in the execution.

To insert a break point:

a. position the EDIT CURSOR on the line before the one where the break point is to be inserted

b. press the Execution function key (F2) from the main menu of the IDE Page

c. in the pop-up menu that is opened by the system, choose the Break point function.

To remove a breakpoint, perform the same operations used for the insertion.

4.1.3.6 Bypassing an instruction

To continue running the program when there is an instruction waiting to be completed (example: WAIT FOR, DELAY, etc.), the Bypass command can be used:

a. press the Execution function key (F2) from the main menu of the IDE Page

b. in the pop-up menu that is opened by the system, choose the Bypass function (item 7).

4.1.3.7 Execute a temporary instruction

To execute an instruction that will not be included in the body of the current program, proceed as follows:

a. press the Execution function key (F2) from the main menu of the IDE Page

b. in the pop-up menu that is opened by the system, choose the Immediate function

c. in the central menu there is the Statement function that opens a dialogue window to insert the instruction required.

d. Select the instruction and press ENTER to confirm the choice; the instruction can also be typed in, using the Alphanumeric keypad

e. to confirm the execution of the chosen instruction, press 'OK'.

4.1.4 Saving the program and closing IDE

To save the.COD and .VAR of the program open in IDE:

a. press the Prog function key (F1) from the main menu of the IDE Page

b. in the pop-up menu that is opened by the system, choose the Save (cod,var) function (item 2). The saved program remains open in IDE.

To save all the modified programs in IDE environment:

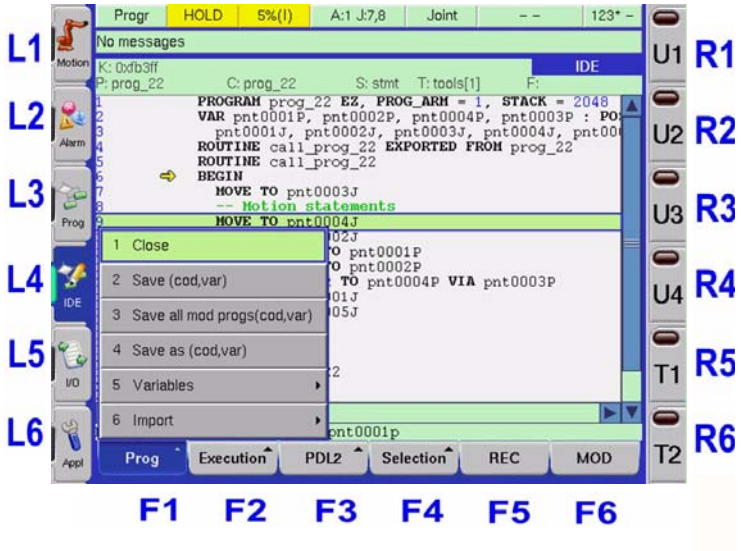
a. press the Prog function key (F1) from the main menu of the IDE Page

b. in the pop-up menu that is opened by the system, choose the Save all mod progs(cod,var) function (item 3).

To close the IDE environment:

a. press the Prog function key (F1) from the main menu of the IDE Page

b. in the pop-up menu that is opened by the system, choose the Close function (item 1).



4.2 Executing the program in automatic mode

It is advised to only run programs that after thorough checking (debugging), are considered able to operate correctly.
If the program has not been debugged before execution, it could cause personal injury and damage to the equipment.

a. Check the following points before running programs in automatic mode:
   - the program is to have been thoroughly checked (debugged) and deemed able to operate correctly
   - there must be no-one in the working area, and this is to be without obstructions
   - all the safety protections are to be installed and in perfect working order
   - all other specific conditions relating to the application and the installation are to be thoroughly checked and deemed able to operate correctly

b. press the HOLD (yellow) key

c. set the modal selector switch to AUTO (or REMOTE)

d. set the required speed override and step mode, enter any useful break points;

e. press the START (green) key to run the program (in the step mode set).

In IDE environment, the execution of the program that is currently open can be observed in the Editor area. For further information see Displaying the currently being executed instruction.

5 PDL2 Programs - Basic concepts

Programs for the C4G Control Unit are written in PDL2 Programming language.

– PDL2 program Structure;

– HOLD/NOHOLD Attribute.

5.1 PDL2 program Structure

PDL2 Programs have the following structure:

```
PROGRAM name <attributes>
  <import statements>
  <constant, variable, and type declarations>
  <routine declarations>
BEGIN <CYCLE>
  <executable statements>
END name
```

5.2 HOLD/NOHOLD Attribute

We can classify them into two main categories, depending on the assigned holdable/non-holdable attribute:

– ‘holdable’ Programs (i.e with HOLD attribute) - their execution is controlled by the HOLD (yellow) and START (green) keys; a Program including motion statements MUST be a ‘holdable’ Program. For further information about how to use HOLD and START keys, see Per ulteriori informazioni riguardanti l’uso dei tasti HOLD e START, vd. par. Other colours keys on page 1-29.

– ‘non-holdable’ Programs (i.e with NOHOLD attribute) - they are usually used for process controlling. They CANNOT include motion statements, even if they are allowed to use positional variables as well, for other purposes.

The default attribute is HOLD; if it is wished to write a ‘non-holdable’ Program, it is needed to explicitly insert the NOHOLD attribute.

6 Programs structure overview

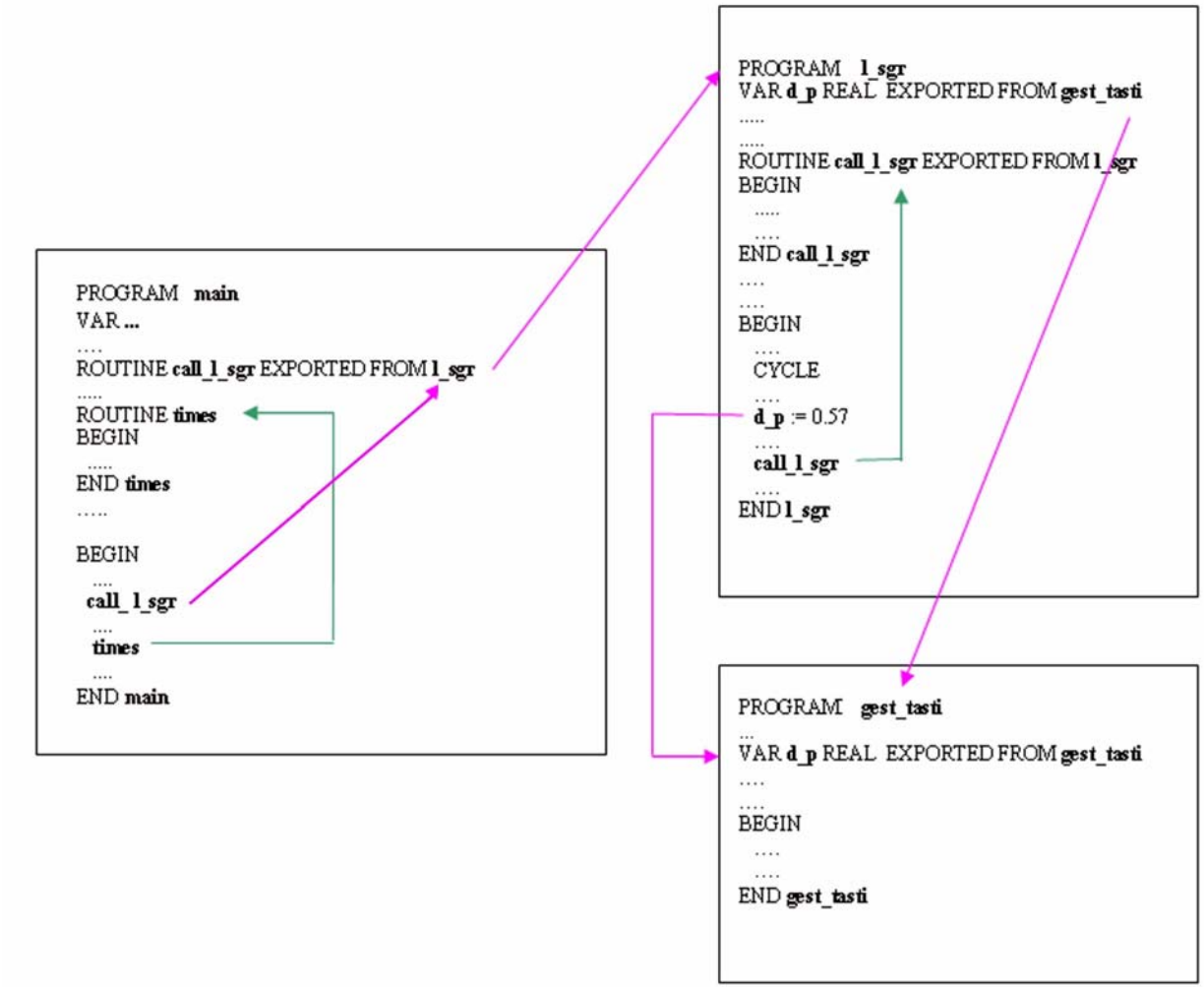
An example follows of a PDL2 program that calls up the LOCAL routine, or is IMPORTED from external programs or defined in it and declared EXPORTABLE:

```
PROGRAM main
-- local variables declaration
VAR pnt0001j, pnt0002j, pnt0003j, fuori_ingombro : JOINTPOS FOR ARM[1]

-- declaration of routines exported from other programs
ROUTINE call_l_sgr EXPORTED FROM l_sgr -- imported routine from 'l_sgr'
ROUTINE help EXPORTED FROM gest_tasti -- imported routine from 'gest_tasti'

-- local routines declaration to 'main' program
ROUTINE selez
VAR lun : INTEGER -- local variable to routine 'selez'
BEGIN
   .......
   help -- calls 'help' routine, imported from 'gest_tasti'
END selez

ROUTINE times -- local routine to program 'main'
BEGIN
   .....
END times

-- beginning of 'main' program body
BEGIN
   ...........................
   CONDITION[1] :
     WHEN $DIN[5]- DO
     times   -- calls local 'times' routine
   ENDCONDITION
   ...........................
   call_l_sgr -- calls imported 'selez' routine (which calls imported routine 'help')
   ...........................
END main
```

An example follows of ‘l_sgr’ program definition to which ‘main’ program refers:

```
PROGRAM l_sgr EZ, STACK = 2048
-- dichiarazioni di variabili locali al programma ‘l_sgr’
VAR inizio_ciclo, pnt0003p, p1, p2, p3, p4 : POSITION

-- dichiarazione di variabili importate da ‘gest_tasti’
VAR d_p, d_s_finitura, altezza_bordo : REAL EXPORTED FROM gest_tasti

-- dichiarazione di variabile locale al programma ‘l_sgr’
VAR cont : INTEGER

-- dichiaraz. di routine definita nel programma ‘l_sgr’ed esportata all’esterno
ROUTINE init_motion EXPORTED FROM l_sgr
ROUTINE init_motion -- definizione della routine ‘init_motion’
BEGIN
   ..............
END init_motion

-- dichiaraz. di routine definita nel programma ‘l_sgr’ed esportata all’esterno
ROUTINE call_l_sgr EXPORTED FROM l_sgr
ROUTINE call_l_sgr -- definizione della routine‘call_l_sgr’
BEGIN
   ..............
   init_motion -- chiamata alla routine ‘init_motion’ definita in ‘l_sgr’ ed esportata all’esterno
   ..............
   MOVE LINEAR TO pnt0003p
   ..............
END call_l_sgr

-- corpo del programma l_sgr
BEGIN
   CYCLE
   d_p := 0.57 -- variabile importata da ‘gest_tasti’
   call_l_sgr -- chiamata alla routine ‘call_l_sgr’
   WAIT FOR $FDIN[21]
END l_sgr
```

A diagram follows about the connections among the listed above programs ::



The **green** arrows indicate the LOCAL connections; the **fuxia** arrows indicate the connections between programs.

When a program is developed in IDE environment, the user is helped to insert the program lines (declarations, program body lines). For further information, please refer to the Appendix.

7 Life cycle of a PDL2 Program

- Creating and developping a Program,
- Saving programs to external storage devices.

7.1 Creating and developping a Program

The recommended steps for program development are as follows:

a. write the program (using a developping environment - e.g. IDE), i.e. write code, define new routines, define variables and teach motion position values;

b. execute the program and check that it is correct, using the Debug commands (existing in the developping environment). With these commands, you can modify execution mode (one instruction at a time, continuously, etc.), examine the values of the variables, modify the code of the program (including the declaration part of the variables). You can also tests movements at low speed;

c. save the variables (a .VAR file is then created) and the code (a .COD file is then created) of the developped Program. Saving is automatically executed in the User Directory (UD:).

d. once you are satisfied with program functioning, try to run it in a mode that increasingly resembles effective functioning in production (developping environment). When the modal selector switch is set on T2 (AUTO-T) or AUTO, the motion statements are executed at real speed (in PROGR state, they are executed at low speed for safety reasons);

e. now the Program is ready to be executed in automatic modality. Save code and variables again, by means of the saving command available in the used developping environment.

f. Activate the developped Program from within the Prog Page on the Teach Pendant, and press the START (green) key.

7.2 Saving programs to external storage devices

After having created, developped and saved a Program, it is required to save it to an external device (disk-on-key, PC, etc.) too.
To perform such an operation, use the Send to (FC) command, available in the Files Page on the Teach Pendant (or the FilerCopy command from within WinC4G environment).
Using the command, specify the destination device: either disk-on-key (XD:) or another device (COMP:).
If the user wishes to save/restore the whole content of the User Directory (UD:), the Backup e Restore procedures have to be used.

Basic Training Course - C4G USE AND PROGRAMMING - Third Day

- **Basic information about memory structure and file types**

  1 - System memory structure
  2 - File types

- **Interface on PC to C4G Control Unit and off-line programming**

  3 - WinC4G Program
  4 - Off-line programming

- **Communications between C4G Control Unit and external devices**

  5.1 - Communication between Control Unit and peripherals
  5.2 - Communication by means of digital I/Os
  5.3 - Programs for Fieldbus and I/O configuration
  5.4 - PDL2 statements for communications between Control Unit and external devices

1 System memory structure

C4G Control Unit provides several storage devices to save, load, execute the user Programs:

– Internal storage Devices,

– External storage Devices.

1.1 Internal storage Devices

Internal storage devices are the ones provided by the C4G Controller Unit. Moreover:

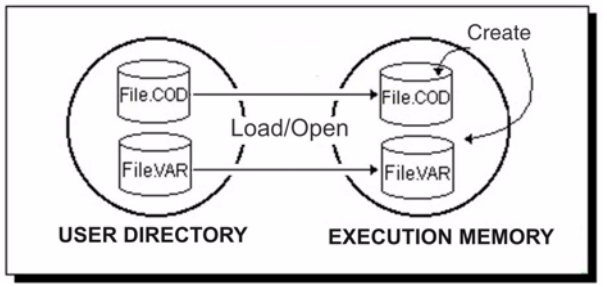
– Execution memory

– User Directory (UD:)

1.1.1 Execution memory

The execution memory includes all the Programs which are displayed in the Prog Page. screen. Thus, .COD and .VAR files only.

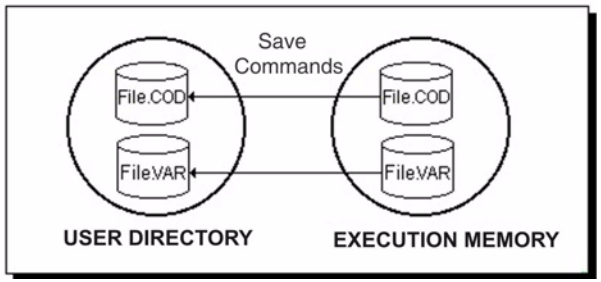
When the user issues either Load... or Open (IDE) or Create commands for a Program, its .COD and .VAR are copied into the execution memory.



1.1.2 User Directory (UD:)

The User Directory can include any kind of files.

After either modifying or creating programs in Execution memory, in order not to lose their associated data, the user must copy them to the User Directory (UD:).
To do that, issue save commands (Save (MS + MS/C), Save data (MS), Save code (MS/C), Save variable as (MS/A)).



1.2 External storage Devices

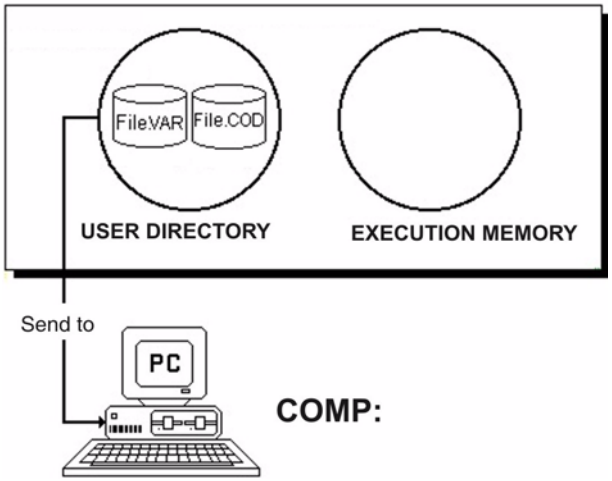
These devices are used to create a Save Copy of files included within the User Directory (UD:) (mainly if they are User Programs). The file transfer can be executed towards:

- PC (and associated devices) - COMP:
- Disk on key - TX: and XD:

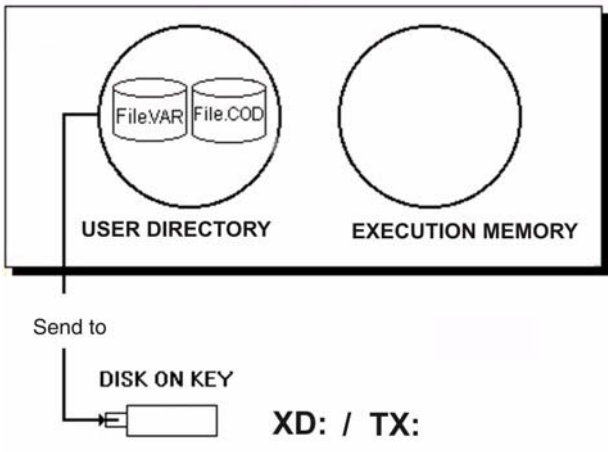
1.2.1 PC (and associated devices) - COMP:

It is possible to copy files from User Directory (UD:) to a PC connected to the Control Unit, with WinC4G running on it (see par. 3 WinC4G Program on page 3-5).

Issuing Send to (FC) command from within the Files Page and specifying COMP: as the destination, the indicated files are copied to the PC selected directory.



1.2.2 Disk on key - TX: and XD:



It is possible to copy files from the User Directory (UD:) to a disk on key device, mounted on the USB port of the Teach Pendant.

Issuing Send to (FC) command from within the Files Page and specifying either TX: (if the TP USB port is used) or XD: (if the optional Controller Unit port is used) as the destination, the indicated files are copied to the disk-on-key.

2 File types

C4G Control Unit handles the file types listed in the following table.

Extension	Description	Backup extension (where applicable)
.ACT	Single file containing list of actions executed on Controller (ASCII format of .LBA files)	.BKA
.LBA	Binary file containing list of actions executed on Controller	.BKB
.COD	PDL2 program code, in binary format. It can be edited in IDE environments, Program Edit, Memory Debug	.BKC
.LBE	Binary file containing list of errors detected on Controller	.BKE
.LOG	FileLOG file, for example ACTION.LOG and ERROR.LOG	.BKG
.LSV	ASCII version of .VAR file	.BKL
.XML	eXtensible Markup Language type file, used to define additional syntaxes on Controller	.BKM
.PDL	ASCII version of a PDL2 program	.BKP
.UDB	User Data Base type file that contains all permitted User Profiles	.BKU
.VAR	Binary file containing variables of a program	.BKM
.VPS	Source file for VP2.Builder	.BKW
.LVP	ASCII version of VP2 file	.BKX
.ZIP	Compressed file	not foreseen
.C4G	Binary file containing system configuration. It can be loaded or saved in the system memory using Config. Function, from Setup page (and the corresponding ConfigureLoad and ConfigureSave commands from WinC4G and TP-INT; for further information, see CONFIGURE branch, Load menu and Save menu).	.BK4

3 WinC4G Program

The Winc4g program is a Windows style interface on PC to C4G Control Unit. It contains several functions:

The connection between PC and C4G Control Unit, to use WinC4G program, can be implemented in one of the following ways:

Type of connection	Characteristics
Point to point Ethernet	– Connecion by twisted Ethernetcable category 5 between Ethernet port 10/100 on Personal Computer and Ethernet port on SMP+ (connector X5). – To make the connection it is necessary to open the cabinet door.
Ethernet to local network	– Connection by straight Ethernet cable category 5 tbetween Ethernet port 10/100 on Personal Computer and Ethernet port on HUB / Switch of the local network. It is also possible to access via Router or Internet connections (check the limits set by the network administrator). – To make the connection the Control Unit has to be connected to the Ethernet network and be accessible.

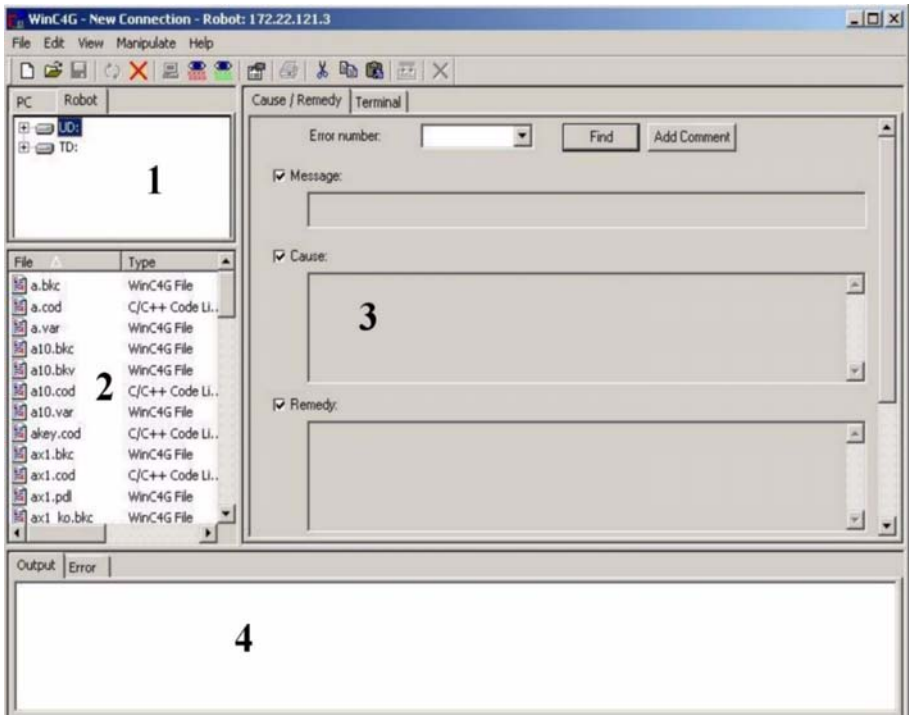
WinC4G provides some functionalities, such as:

– display of files installed on the robot;
– possibility to translate in executable form (.COD) and edit them;
– errors search and display;
– possibility to open a Terminal window to directly forward commands onto the Control Unit;

To activate it, just select Winc4g.exe by clicking on the file or its correnponding link.

3.1 WinC4G User Interface

The WinC4G interface consists of four main panels each of which can contain one or more windows with information for the user:



- (**1**) Directories Panel
- (**2**) Files Panel
- (**3**) Tools Panel
- (**4**) Output Panel.

3.1.1 Directories Panel

The Directories panel shows the structure of the PC and the Control Unit directories (for the latter, if the connection via Ethernet is active).
Only the directories on the devices are displayed, NOT the files they contain.
The Directories panel has one or two windows according to whether the connection to the Robot Control Unit is active or not.The display can be updated using the "Display Update list of files" command.

3.1.2 Files Panel

The Files panel shows the list of the files contained in the device or folder that has been selected on the Directories panel. For each file the information is shown relating to the name, type, size and date of change. The items can be set in order according to one of these categories.

As from system software version 3.20, further information can be viewed (Properties) relating to the files, clicking with the mouse right-hand key and selecting “Property Notes”. Standard attributes (Property) associated to the files are the following:

3.1.3 Tools Panel

The Tools panel is the main panel of the application. It is in this area that most of the information is displayed. Several application functions may be present on this panel. The possible windows are:

– Terminal window
  it is the direct interface towards C4G (similar to the TP-INT Page); it displays the information that is on the Control Unit video

– File Translation window
  allows the user to display and translate programs, from ASCII format (.PDL) to internal format (.COD);

– Cause and Remedy window
  allows the user to search for both an error cause and the associated remedy to fix it;

– Errors and Actions window
  allows the user to display and sort errors and actions issued by the Control Unit;

– Files manipulation window - provides several operations to check and/or convert variables values. For all the Manipulation menu functionalities, the input data to be ALWAYS specified, are as followsr
  - System file: name of the .C4G file.
  - Tool file: name of the .VAR file containing the values of the tools used by the Program (for example UD:\DATA\TT_TOOL.VAR).
  - Frame file: name of the .VAR file containing the values of the frames used by the Program (for example UD:\DATA\TU_FRAME.VAR).
  - Variables file: name of the .VAR file corresponding to the user Program to be converted.
  - Program file: name of the .COD file associated to the user Program to be converted.

To close a window on the Tools Panel, send the Close command from the File menu.

3.1.4 Output Panel

The Output panel displays the messages that an application generates to the user. The main use is to show the result of the program files translation, such as .PDL and .LSV, including the translation errors and messages coming from the file manipulation operations.
Clicking twice on an error, opens the file on the line where the error has been found.

There are also one window to display FTP Server messages and another one for the chat, when WinC4G is connected either as Remote or Proxy.

To open this Panel, use "Show Output Window" in the View menu; to close it, use "Close Output Window" from the same menu.

4 Off-line programming

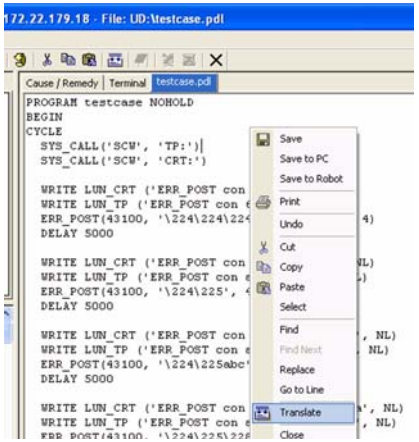
From within the WinC4G environment the user is allowed to edit PDL2 language Programs, even when the PC is not connected to the C4G Control Unit (off-line programming).

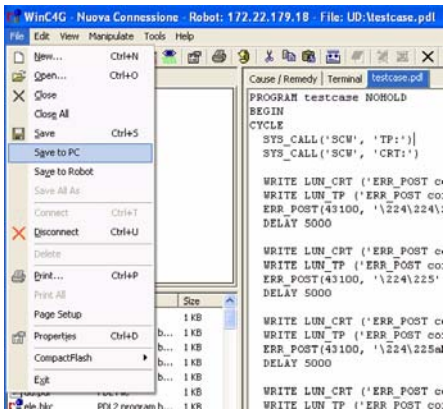
Proceed as follows:

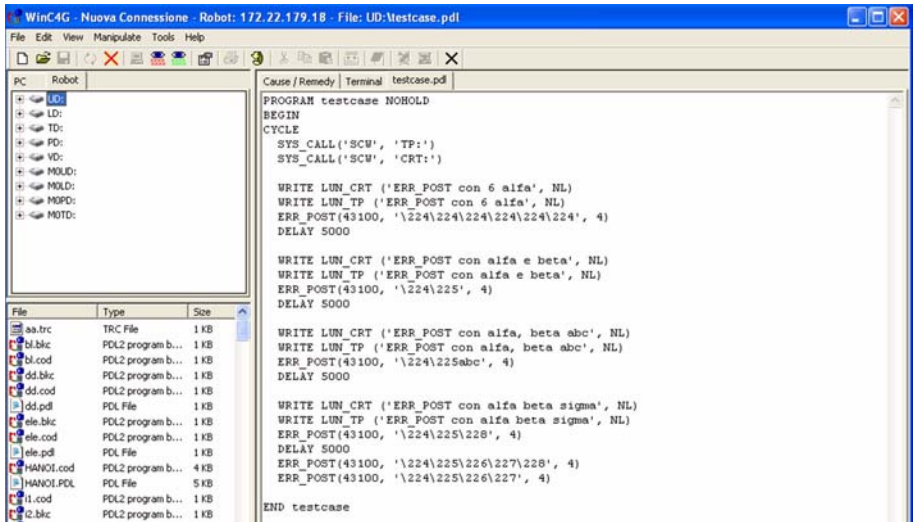
a. **Open the file (new or already existing))** - this means to open a new window in the Tools Panel, by means of either the New command, from File menu, or opening an already existing file (either Open command, from File menu, or double clicking on the file name - see here beside figure).

b. **Editing some modifications or the whole program**.

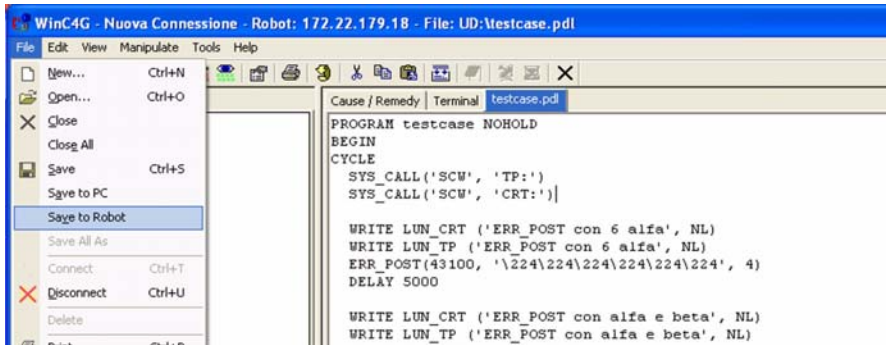
c. **Translate into internal format**- to run translation, either choose the Translate command from Modify menu, or right click and choose Translate command (see here below figure on the left).
If the program is new, it is needed to save it first (Save command, from File menu - see here below figure on the right) .







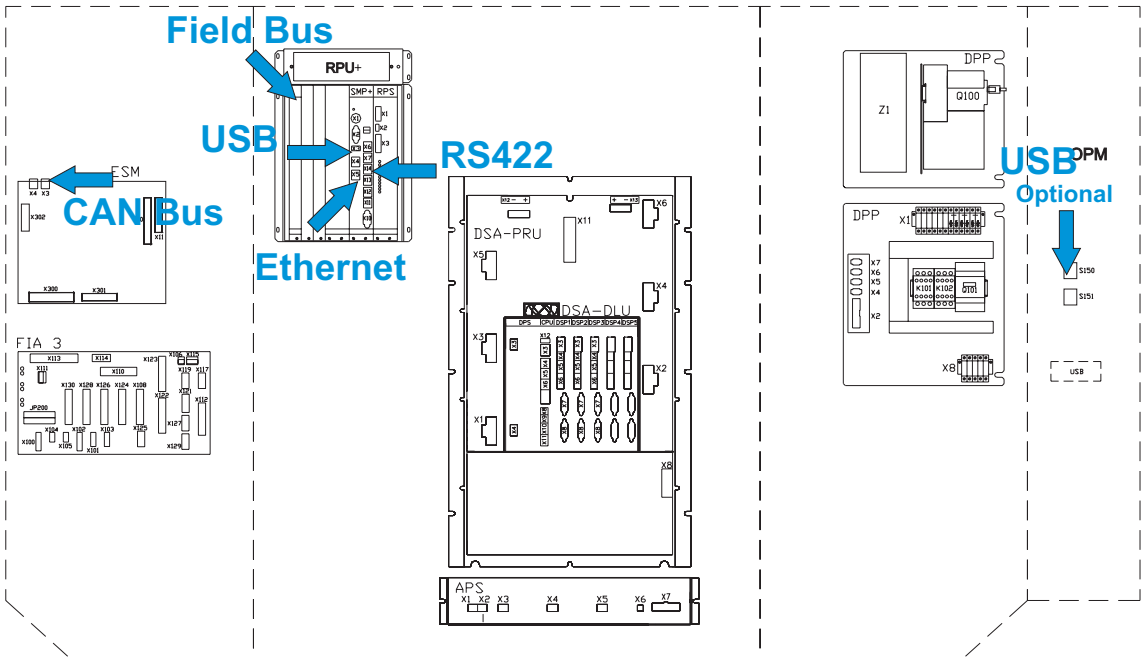
d. **Copy the program to the Control Unit** to execute it - select Save to Robot command, from File manu (see here below figure); obviously, to be able to copy the file, Control Unit and PC must be already connected.



5 Communication between Control Unit and external devices

- Communication between Control Unit and peripherals
- Communication by means of digital I/Os

Communication Ports layout



5.1 Communication between Control Unit and peripherals

C4G Control Unit is provided with some basic features and options which allow to communicate with the following ports:

– Ethernet port,

– Serial port,

– USB port.

5.1.1 Ethernet port

The C4G Control Unit is provided with the Ethernet network connection. The Ethernet networks ensure a high speed data exchange where economic connections are required to a local communications system with intense traffic and high peak speed. A station error does not jeopardise the functioning of the others.

The connection on the Ethernet network is available on connector X5 (type RJ45) of the RPU module CPU SMP+. The Ethernet network is automatically activated at power-on.

5.1.2 Serial port

The C4G Control Unit has as part of the standard configuration an RS422 serial port on connector type RJ45 installed on the SMP+ board. The port is isolated from the other RPU+ circuits.

It is also possible to obtain an RS232 serial port, using a suitable converter (not available as an option).

5.1.3 USB port

The C4G Control Unit has as standard the following USB ports:

– USB port installed on SMP+ board. Remote control of the USB connection on the CPU SMP+ on the cabinet right door is possible. The option contains the cable to connect to the USB port of the CPU SMP+, the connector and the IP 54 casing to fasten on the cabinet door. The connection allows the use of devices compatible with USB 1.1 such as Disk On Key (optional).

– USB port on the Teach Pendant (see par. 6.4 USB port on page 1-32 - Basic Training Course **First Day**)).

The available Communication Ports on C4G Control Unit are summarized in the following table.

C4G port		Function		Device
Ethernet port	**Ethernet Port** [X5]	Interface towards WinC4G Program (*)	**basic**	[**NET0:**]
		FTP		[**NET1:**] e [**NET2:**]
		Socket on Ethernet TCP / UDP protocol with PDL2	optional	[**NETT:**]
Serial port	**RS422** on SMP+ board [X7]	Generic serial line	**basic**	[**COM1:**]
USB port	USB port on SMP+ board [X3] or on cabinet door [USB] (**optional**). See par. USB port (optional feature) on page 1-6 of Basic Training Course **First Day**	Storage Disk On Key	optional	[**XD:**]
	USB port on the Teach Pendant Vd. par. 6.4 USB port on page 1-32 of Basic Training Course **First Day**	Storage Disk On Key	**basic**	[**TX:**]

(*) Regardless the channel the PC is connected to, the user must always refer to WinC4G using the name COMP: (see par. 1.2.1 PC (and associated devices) - COMP: on page 3-3).

For further information, refer to par. 9.1.4 COMMUNICATION Devices on page 9-3 in **PDL2 Programming Language** manual.

5.2 Communication by means of digital I/Os

- Digital I/Os available on the Control Unit, ready to be used on the Robot
- Optional digital I/Os connected via CANBus
- Optional digital I/Os connected through Fieldbus
- Fieldbus Connections (CAN, DeviceNet, Profibus-DP, Interbus-S) between C4G and Robot
- Programs for Fieldbus and I/O configuration
- PDL2 statements for communications between Control Unit and external devices

5.2.1 Digital I/Os available on the Control Unit, ready to be used on the Robot

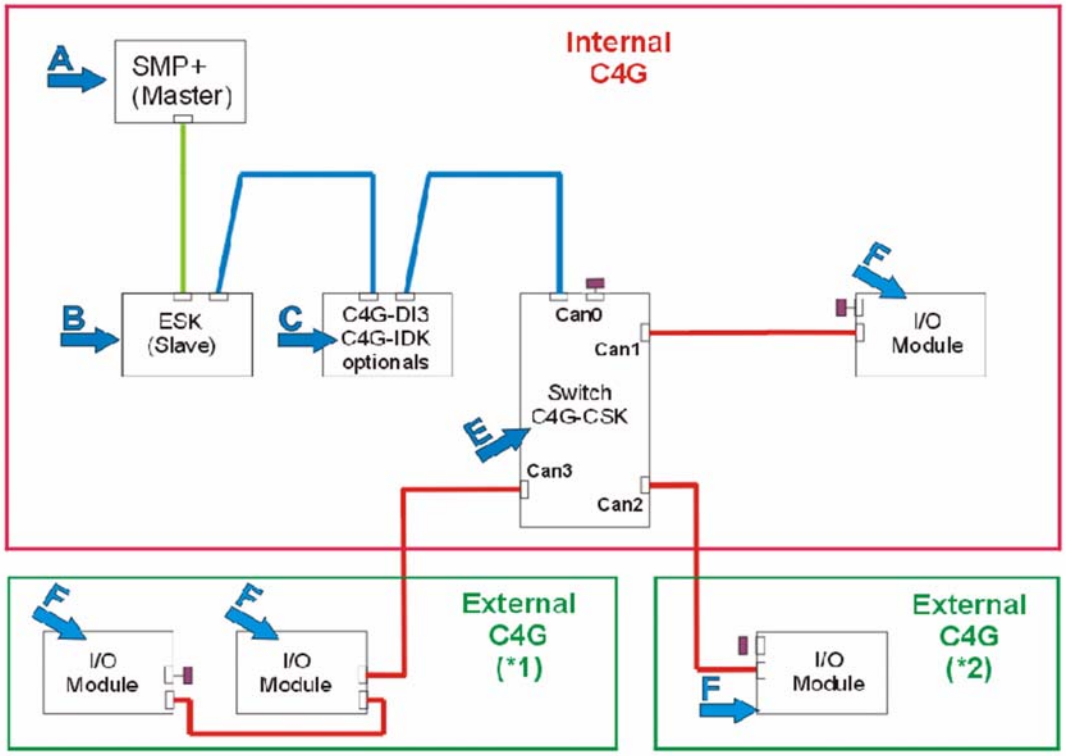
For minor automations on board the robot, the Control Unit has parallel digital inputs/outputs (I/O). There are 2 predefined inputs (air alarm and flange alarm) and 4 inputs and 2 outputs that can be used at will by the user.
These I/Os are available on the DSA-CPU and their signals are connected to the Robot base by means of the X10 connector of Robot signals. Depending on the Robot configuration, they are available from axis 1 to axis 3 (see manuals **Smart NS - Transport and Installation** - chap. **DSA connector and I/O cable**).

To be able to use I/Os in PDL2 programs, it is needed to configure them by means of the IO_TOOL Program - I/O Configuration in VP2.

5.2.2 Optional digital I/Os connected via CANBus

For applications which require some more I/Os, Comau System is already provided with a CAN port on the ESK module, to which connect slave modules compatible with the CAN Open protocol, such as signals Converters, CAN Switch and I/O modules.
Since the native CAN network is inside the C4G Controller, it is enough to add the desired I/O modules (COMAU options) and simply connect them by means of an Ethernet cable.

In the here below figure, a CAN network example is shown.



A: Master network node (Control Unit CPU SMP+)
B: System Slave network node (ESK, electronic safety module)
C: Optional I/O Slave module, type C4G-DI3 and/or C4G-IDK
E: Switch CAN module type C4G-CSK, terminating on Can1, Can2 and Can3 branch.
F: Optional I/O Slave module type C4G-IDK and/or compatible commercial module

Connection inside C4G cabinet, present in basic system

Connection with other slave nodes inside C4G cabinet, implemented according to requirements

Connection with other slave nodes outside C4G cabinet, implemented according to requirements

(*1) It represents an external installation (e.g to control rotating tables or other fixtures)

(*2) It represents an istallation on Robot board (with one or more I/O modules to control gripper or other fixture on robot wrist).

It is possible to choose between the following I/O modules, connected to the RPU module via CANBus network:

– C4G-DI3module (optional) for installations on the base of the cabinet and connection via multipolar connector

– C4G-IDK modules (optional) for internal and external connection to the cabinet, in protection enclosure, pin connections.

Each module on a CAN network is to be configured with a univocal address. The addresses of I/O modules type C4G-IDK and C4G-DI3 can be changed by means of 2 rotary selector switches SW1 and SW2. The address consists of 2 digits. The digit with least significant bit is selected on SW1, the most significant bit on SW2.

To be able to use I/Os in PDL2 programs, it is needed to configure them by means of the IO_TOOL Program - I/O Configuration in VP2.

5.2.2.1 C4G-DI3module (optional)

With the C4G-DI3 boards parallel I/Os can be used to interface with the applications. The boards have 16 digital input points and 16 digital I/O points that can be configured as desired.
These modules are connected to the RPU module, via CANBus network. They are useful to add parallel I/Os INSIDE the Controller Cabinet.

For the connectors pin-out, see C4G-DI3 module connector pin-out.

5.2.2.2 C4G-IDK modules (optional)

The C4G-IDK module provides 16 digital input points and 16 digital input/output points that can be configured as desired in IP20 enclosure, for use inside the Control Unit cabinet. Each I/O point is monitored by a yellow LED.
This module can also be used OUTSIDE the cabinet protected by an envelope: interpose a switch to galvanically insulate C4G from outside (COMAU option Switch CAN C4G-CSK module (optional) is available), and use an Ethernet cable.

For the connectors pin-out, see C4G-IDK module connector pin-out.

5.2.3 Optional digital I/Os connected through Fieldbus

For applications requiring more I/Os, in the case in which a Fieldbus board different from CANBus is needed (e.g. due to factory/design standars), the Comau System requires the specific Bus board, the suitable cables and the wished I/O modules (not provided as Comau options but available on the market).

All the field bus boards have two separate channels:

– Slave for remote control by the cell management PLC,

– Master to control local I/Os to robot and to application.

The allowed Fieldbus types are as follows:

– DeviceNet,

– Profibus-DP,

– Interbus-S.

After the installation of the chosen board in the RPU module, it is needed to properly configure it by means of the FB_TOOL program for configuration of Fieldbus.

5.2.4 Fieldbus Connections (CAN, DeviceNet, Profibus-DP, Interbus-S) between C4G and Robot

The connection between System and digital I/Os is implemented by means of a Multibus connector and a Multibus cable (see the figure on the right):

– it is It is the Master channel of the field bus board (A) that connects to the I/O slave modules on-board robot. For the CANBus, please start from the last CAN node (Switch)

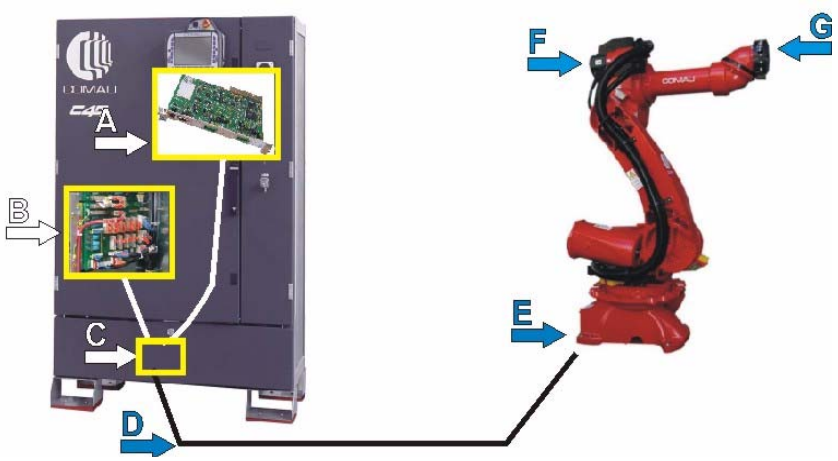
– The board Master channel output connector and the 24 Vdc power supply taken from connector X137/FIA (B) go to the Multibus connector X93 (C). To this purpose special optional Master cables are used, that differ according to the network they support.

– On Multibus connector X93 (C) at the cabinet base Multibus cable (D) can be connected to carry the signals and the power supplies to the robot base (E).

Depending on the purchased outfitting, the robot is provided with:

– an Internal or external Multibus cable and

– a Multibus connector on axis 3 (F) or at wrist (G).



5.2.4.1 Multibus connector

The Multibus connector is a 17-pin multipolar type.
The Control Unit and the Robot have only one connector because they are considered as the two ends of the chain of devices connected to the Multibus line. To allow signals to transit, any intermediate devices have to be fitted with an input connector and an output connector.

5.2.4.2 Multibus cable

The Multibus cable is a multipolar cable to connect the field bus network signals and the 24 Vdc required to supply the remote I/O modules.

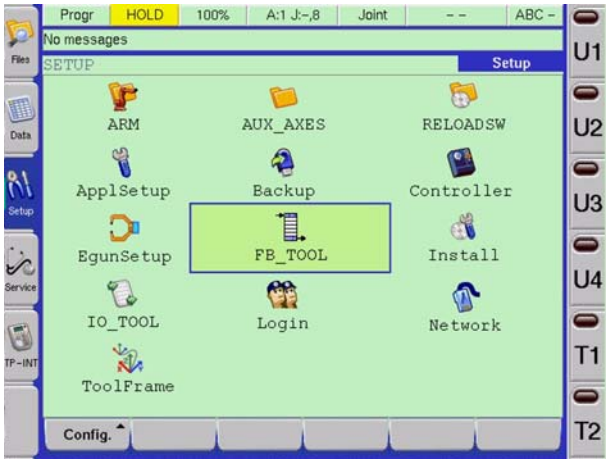
The Multibus cable inside the cabinet is to directly connect the Fieldbus boards; the external one, either towards the robot or inside the robot, consists of two Multibus connector at the each end and the link-up between the two connectors is direct (pin to pin).

For the pin-out and electrical characteristics of both the Connector and the Cable, please refer to the Applications Installation manuals - , Chapter “Connectors and multibus cable” and the robot Circuit Diagrams.

5.3 Programs for Fieldbus and I/O configuration

- FB_TOOL program for configuration of Fieldbus
- IO_TOOL Program - I/O Configuration in VP2
- IO_INST Program - I/O configuration from WinC4G.

5.3.1 FB_TOOL program for configuration of Fieldbus



This program, in Visual PDL2, and therefore with an efficient graphic interface, is used for the configuration, viewing and resetting of fieldbus and CAN modules, on the C4G Control Unit.

The FB_TOOL program is always present and available on the Teach Pendant. To use it, select FB_TOOL icon in the Setup page (see the figure on the right).

In the FB_TOOL program Home Page, the user can enter the FIELDBUS environment (by means of **F1** softkey, as shown in the figure on the left).
With this function it is possible to configure, view and reset the Fieldbus boards (DeviceNet, Profibus and Interbus) and the CAN modules.

For detailed information about FB_TOOL program, refer to C4G - Control Unit use manual- Chap. FB_TOOL program for configuration of Fieldbus.

5.3.2 IO_TOOL Program - I/O Configuration in VP2

This program is in Visual PDL2, therefore with an efficient graphic interface, it is used for the configuration, viewing and resetting of input/output, DSA, HAND relating to the Arms, on the C4G Control Unit.

The FB_TOOL program is always present and available on the Teach Pendant. To use it, select IO_TOOL icon in the Setup page (see the figure on the right).

In the IO_TOOL program Home Page, (see the figure on the left) the user can enter the following environments:

- I/O ports(F2)
- DSA (F3)
- HAND (F4)
- Save (F5)
- Close (F6)

For detailed information about IO_TOOL program, refer to C4G - Control Unit use manual- Chap. IO_TOOL Program - I/O Configuration in VP2.

5.3.3 IO_INST Program - I/O configuration from WinC4G

This program is used for the configuration, viewing and resetting of the configuration itself, of the inputs/outputs, of the field buses and of the CAN modules on the C4G Control Unit.

To install the program it is necessary to:

a. connect the PC to the C4G Controller and run the WinC4G program.

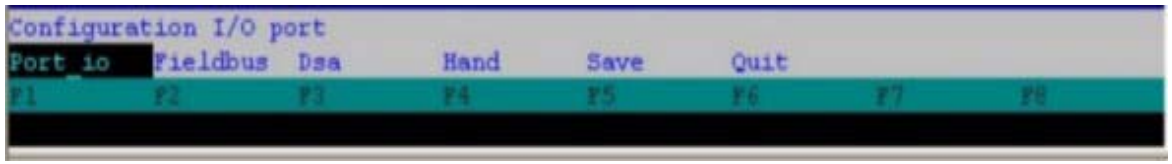
b. from WinC4G select the directory into which the files have been transferred.

c. run the Filer Utility Install IO_INST command from WinC4G. The corresponding files will be automatically copied from PC and the program will be run on WinC4G.

The program can only be run on WinC4G (therefore on PC) and not on the Teach Pendant.

This program operates in a manner similar to the command language menu.

The menu is displayed on the bottom part of the WinC4G Terminal window (see the start menu in the figure here below). The selections can be made through either the function keys or the cursor keys confirmed by the ENTER key.



For detailed information about IO_INST program, refer to C4G - Control Unit use manual- Chap. IO_INST Program - I/O configuration.

5.4 PDL2 statements for communications between Control Unit and external devices

PDL2 provides some statements, predefined variables and built-in routines to handle communications between C4G Control Unit and external devices.

PDL2 statements useful to handle simple communications, are the following:

- assignment - assigns either a value or an expression to a variable.
- OPEN FILE Statement - establishes a connection between the program and the device through which I/O operations can be performed (e.g. writing and reading data).
- CLOSE FILE Statement - ends the connection between the program and the device,previously started by the OPEN FILE statement.
- READ Statement - legge dati in Input dal dispositivo specificato verso il programma.
- WRITE Statement - writes output data from a program to the specified device.

The available **predefined variables**, to handle communications from within PDL2 programs, belong to the PORT System Variables category (e.g. $DIN: Digital input, $DOUT: Digital output, $SDIN: System digital input, etc.).

The available **built-in routines** to manipulate bits, from within PDL2 programs, are the following:

- BIT_CLEAR Built-In Procedure - clears a bit of either an INTEGER variable or a port
- BIT_SET Built-In Procedure - sets to 1 a bit of either an INTEGER variable or a port
- BIT_TEST Built-In Function - checks whether the specified bit of either a variable or a port is 1 or 0.

For any further information, refer to APPENDIX1 - par. 2 Basic PDL2 Statements on page 6-6.

Basic Training Course - C4G USE AND PROGRAMMING - Fourth Day

- Data saving Procedures
  - 1 - Backup e Restore
  - 1.1 - Backup and Restore commands from Teach Pendant
  - 1.2 - Backup and Restore commands from WinC4 program
  - 1.3 - View currently existing Savesets
- System software handling Procedures
  - 2 - Installing/Upgrading the System Software
  - 2.1 - Installing/Upgrading C4G Software
  - 2.2 - Installing/Upgrading TP Software
- Basic information for WiTP wireless use
  - 3 - WiTP wireless Pairing and Unpairing
  - 4 - System Restart with WiTP wireless
  - 5 - Hints for using the WiTP wireless

1 Backup e Restore

BACKUP command - Saves the wished files to the specified device.

RESTORE command - Restores the files, previously saved with a Backup.

HINTS FOR BACKUP AND RESTORE USE

It is recommended to follow the listed below hints, in order to better understand the difference between executing a simple file copy (FILER COPY (FC) command) and performing well organized Backup/Restore operations:

- **always verify the execution memory and the UD: device to be consistent (no unsaved programs and/or data files) before going on with any Backup or Restore operation.**
- **Always execute a ConfigureSaveAll command (Save the configuration file (CSA) in the Setup page) before issuing a Backup command; always execute a ConfigureLoadAll (Load the configuration file (CLA) in the Setup page) after a Restore operation, in order to restore the same working conditions, based on .C4G file, after Restart.**
- **Schedule periodic Backup operations, during the night, if possible.**
- **Compress the backup in a .ZIP file; execute the restore operation decompressing the .ZIP file.**

The simplest and most efficient method is Backup and Restore commands from Teach Pendant: Backup of Saveset (FUB/S) command and Restore of Saveset (FUR/S) command. They use SAVESETs which have been previously defined in the Setup page.

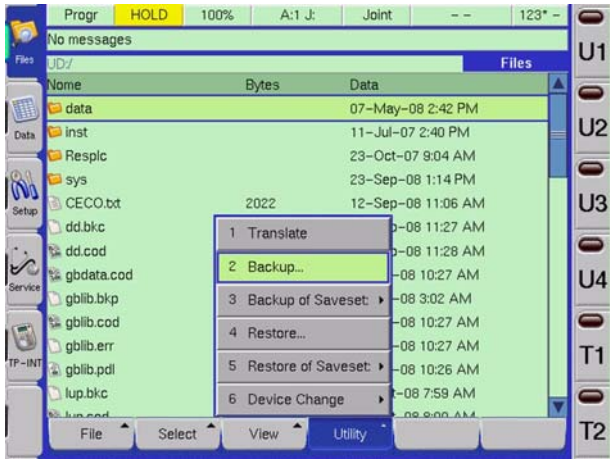
SAVESET

Each Saveset is identified by a Name and includes device, directory (and sub-directories, if needed) and files information, involved in the backup/restore command.

To view the currently defined Savesets, see par. 1.3 View currently existing Savesets on page 4-5.

1.1 Backup and Restore commands from Teach Pendant

The procedure to issue either Backup or Restore command from the Teach Pendant, is as follows:

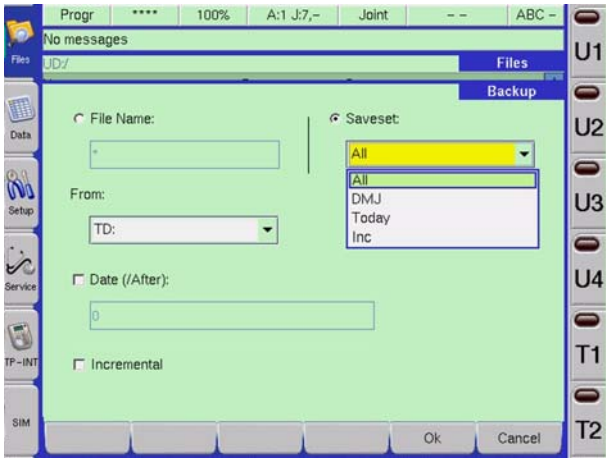


a. select the Files Page,

b. open Utility (F4) menu and select the wished item

b.1 if either Backup (FUB) or Restore (FUR) command is chosen, a screen page is displayed (see the figure here below) in which the user is allowed to specify either a filename or a SAVESET name.
Select the desired choice.

b.1.1 If a filename has been chosen (**File Name** radiobutton is selected - see the figure below on the right), press ENTER to edit the Name field.

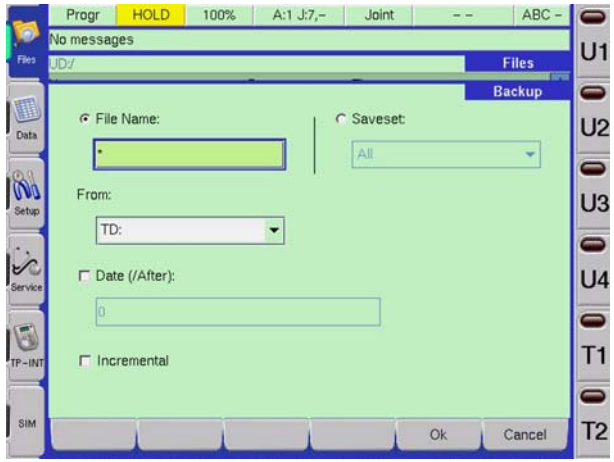


b.1.2 If SAVESET has been chosen (**Saveset** radiobutton is selected - see the figure on the left), open the menu listing the available Savesets and choose the wished one. If desired, some options of the chosen Saveset can be modified.

b.1.3 Press **Ok** (**F5**) to confirm.

b.1.4 The system asks for a confirmation to go ahead with the required operation: answer YES (**Yes**=**F5**, **No**=**F6**) to begin the execution.

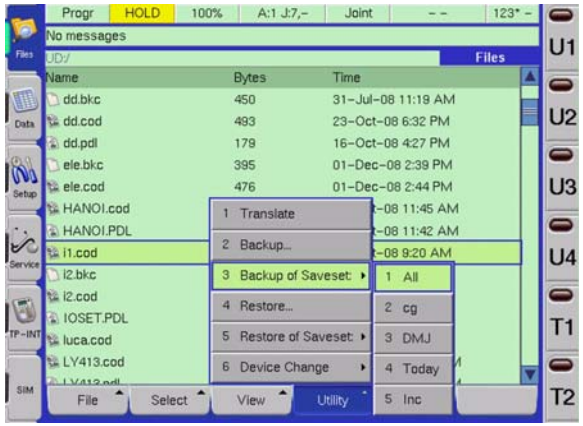
b.2 If, on the contrary, Backup of Saveset (FUB/S) or Restore of Saveset (FUR/S) command is chosen, the existing Savesets list is displayed, without any possibility of modifying them, at this level; the user must choose the wished one (see figure in the nex page) and press ENTER to confirm.



b.2.1 Insert/confirm the destination device (in case of Backup) or the source device (in case of Restore). Press ENTER to open the devices list.

b.2.2 Choose and press **Ok** (**F5**) to confirm.

b.2.3 the system asks for a confirmation to go ahead with the required operation: answer YES (**Yes**=**F5**, **No**=**F6**) to begin the execution.



BE CAREFUL! It is to be reasserted that if using a SAVESET is wished, it MUST be present in the currently defined Savesets list. See View currently existing Savesets.

1.2 Backup and Restore commands from WinC4 program

To issue Backup and Restore commands on a Personal Computer, use WinC4G Program. In such a case, the use of SAVESETs is strongly recommanded. The default Backup Device is the disk-on-key (**XD:**).

The required procedure is as follows:

a. from within WinC4G Terminal window, issue either ***FilerUtilityBackup (FUB)*** or ***FilerUtilityRestore (FUR)*** command;

b. the user is asked for the File(s) Name(s) and the possible options:

b.1 if simply the filename is wished, just type it

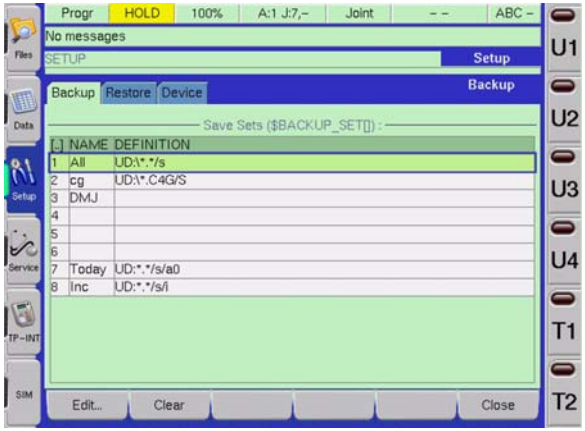
b.2 if it is wished to use a SAVESET, type the '/' character and select the **Saveset** option;

b.2.1 type the wished SAVESET name

c. press ENTER to confirm.

1.3 View currently existing Savesets

On the Teach Pendant, in the Setup page, Backup and Restore sub-pages, the currently defined SAVESETs list is displayed: they are available to perform backup/restore operations. This table can include a maximum quantity of 8 Savesets.



– The string included in NAME column is the Saveset name, which is to be used in the Backup/Restore operations.
– DEFINITION is the description of the Saveset content: device name (e.g. UD:), directory and sub-directories name, file(s) name(s). The use of wildcard '*' is allowed (e.g. file *.cod).

For further information on adding/modifying Savesets, please refer to par. 6.16.5.1 Backup on page 6-169 in the Setup page of **Use of C4G Control Unit** manual.

NOTE - If the user wishes to modify the Device to be used during a Backup/Restore operation, act on Device sub-page of the Setup page.

2 Installing/Upgrading the System Software

To access such commands, enter the Setup page and select ReloadSw icon.

It is then possible to handle the software of the following devices:

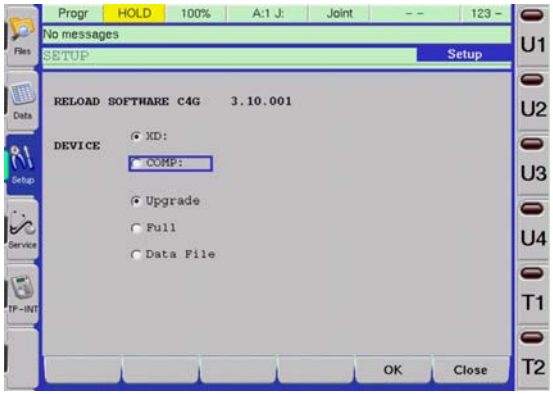
– C4G Control Unit (Installing/Upgrading C4G Software),

– Teach Pendant (Installing/Upgrading TP Software).

Select the corresponding device icon to activate the wished operation.

Fig. Setup - ReloadSW

2.1 Installing/Upgrading C4G Software



On this sub-page it is possible to reload the system software and restart the system.

This operation is especially useful to update system software to a new version.

The user is allowed to load the system software from the following devices: either

– From Disk-On-Key (XD:),
– From PC (COMP:).

From Disk-On-Key (XD:)

a. login with Maintenance profile;

b. insert Disk-on-key in the USB port;

c. select SETUP - ReloadSw - C4G sub-page; see Fig. Setup - ReloadSW;

d. select device XD: ; see the above figure

e. select type of loading: full, upgrade, data file

e.1 Full - which loads the system software and the mechanical Data File; copy the appropriate files and run an automatic Restart to make the newly installed basic software operational. Press OK (F5).

e.1.1 After the Restart, select: loading device, robot family, presence of any auxiliary axes, type of robot to be installed and related serial number;

e.1.2 select the required language;

e.1.3 select the motion programming mode (modal, nodal) which should be modal except for a particular user customization wich means, in this context, reserved

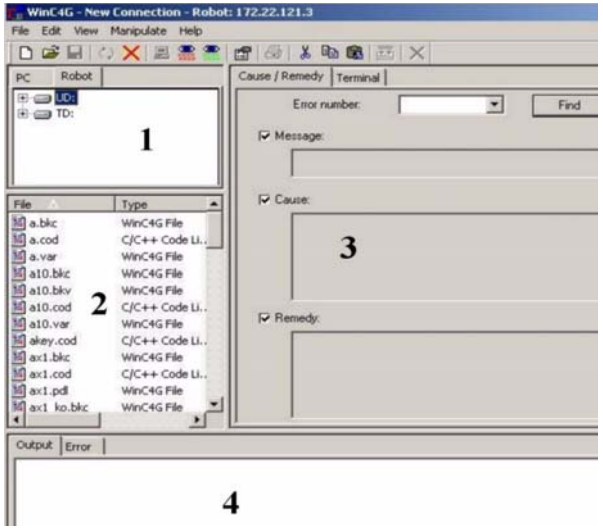
e.1.4 Press the Save key. The system saves in the .C4G file.

e.1.5 Upon termination, press the End key to execute the automatic Restart.

e.2 Upgrade - which loads the new software without changing the system configuration. Press OK (F5).

e.3 Data File - to change the system configuration without updating the operating system. Press OK (F5).

e.4 Reset the Latched alarm if necessary.



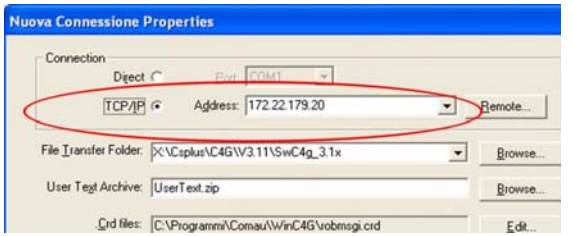
From PC (COMP:)

In this modality, **before** acting on the Setup page, ReloadSw sub-page, it is necessary to perform all the operations described here below, from step a. to step e.

a. Run the WinC4G program on PC; WinC4G has to be connected and the transfer folder has to be correctly set to search for the files to be updated. To do this, proceed with the following steps:

b. on PC select the file transfer directory, (where the software to be loaded is installed); this operation is to be performed from the Directories Panel (no.1 in the figure on the left);

c. set the Control address (for example 172.22.179.20) in the TCP-IP box. This address can be read from the Teach Pendant Home Page; TCP-IP field is in WinC4G Properties Window (see the figure on the right);



d. connect to the Controller with Maintenance Login profile, from the File Connect menu (or icon);

e. to complete the procedure, use the Teach Pendant, as described in previous par. From Disk-On-Key (XD:) on page 4-7, selecting **COMP:** as the loading device, instead of XD:

2.2 Installing/Upgrading TP Software

On this sub-page it is possible to reload the Teach Pendant software, basic software (BSP) and/or User Interface software.

First of all check the BSP version of the Teach Pendant, on the Innanzitutto controllare la versione del BSP del TP sulla Service Page - Sysinfo (SCV), selecting row TP operating system version (BSP). If such a version index is lower than the being installed one, execute the procedure which follows here below, for Teach Pendant BSP loading; otherwise, directly go to the User Interface software loading (SW TP) procedure.

2.2.1 Teach Pendant BSP loading

The procedure to load the Teach Pendant BSP is the following:

a. select SETUP - ReloadSw - TP);

b. select the loading device:
   – if the Teach Pendant is TP4i, select
     • XD: to load from disk-on-key,
     • COMP: to load from WinC4G;
   – if the Teach Pendant is WiTP (wired or wireless), select TX: ;

c. select the Reload button of the TP BSP, and press ENTER to confirm;

d. answer OK to the question asking if the user wishes to reload the BSP of the TP. After loading, a message is displayed to inform the user that the BSP has been successfully copied;

e. DO NOT RE-START FOR WiTP, regardless of whether with or without wire, to avoid problems occurrence. If the Teach Pendant is a TP4i it has to be restarted; the user must confirm the restart command as soon as the system asks for that;

f. go ahead with loading TP software, as described in the next paragraph.

2.2.2 User Interface software loading (SW TP)

The procedure to load the TP User Interface software is as follows:

a. select SETUP - ReloadSw - TP sub-page;

b. select the loading device:
   – if the Teach Pendant is TP4i, select
     • XD: to load from disk-on-key,
     • COMP: to load from WinC4G;
   – if the Teach Pendant is WiTP (wired or wireless), select TX: ;

c. select the Reload button of the TP software and press ENTER to confirm;

d. answer OK (F5) to the question asking if the user wishes to reload the TP software.

After loading, a message is displayed to inform the user that the TP software has been successfully copied to the FLASH memory;

e. restart the Teach Pendant in the following ways:

e.1 if the Teach Pendant is a Wireless WiTP :

e.1.1 unpair the WiTP

e.1.2 restart the WiTP by means of the "Restart TP" button;

e.1.3 shut down the C4G by means of the Controller Main Switch (this is only needed when BSP has been reloaded).

e.2 If the Teach Pendant is a wired WiTP:

e.2.1 if the BSP has been previously reloaded, shut down the C4G by means of the Controller Main Switch, otherwise answer affirmatively to the question about restarting the Teach Pendant.

e.3 If the Teach Pendant is a TP4i:

e.3.1 answer affirmatively to the question about restarting the Teach Pendant;

f. check, in the Service Page - Sysinfo (SCV), that the version of the TP SW, the TP BSP and the C4G software have been properly updated.

3 WiTP wireless Pairing and Unpairing

In the current section we describe the required procedures to properly handle the WiTP wireless Teach Pendant.

To be used, the WiTP wireless pendant must be ON and PAIRED with the Control Unit.

For safety reasons, the WiTP wireless software carries out severe checks on the pairing and requires certain rules to be followed by the user when using it:

- **the WiTP teach pendant can be paired to ONLY ONE Control Unit at a time;**
- **if there are several Control Units with only one WiTP wireless, it is necessary to first unpair the C4G Controller in use, then pair the pendant to another;**
- **when the WiTP wireless is paired to a Controller, it is not possible to place it in a housing (docking station) that is not the docking station of that Controller: if this happens, the pendant is automatically unpaired and the C4G to which it was paired goes into emergency state. This situation can be reset by a procedure of Emergency Unpairing, of the Control Unit to which the WiTP wireless was paired.**
- **However, when the system is working in AUTO mode, the Teach Pendant can be unpaired, if, for example, it is wished to use it on another Controller.**

A detailed description is provided, about the procedures for WiTP wireless pairing and unpairing:

- Pairing procedure between WiTP wireless Teach Pendant and C4G Control Unit
- Unpairing between WiTP Teach Pendant and Control Unit
- Emergency Unpairing.

3.1 Pairing procedure between WiTP wireless Teach Pendant and C4G Control Unit

This procedure, called ‘pairing’, is always necessary in situations where the user wishes to access the C4G Control Unit using a WiTP wireless Teach Pendant.

- **Use the WiTP wireless supplied with the Control Un it, or in any case a teach pendant that is not paired with another C4G Control Unit.**
- **The Pairing procedure can also be carried out with the system in automatic mode.**
- **Always carry out the complete procedure. Do not leave a WiTP wireless without completing the pairing procedure.**

a. Check that the antenna (indicated with letter A in the figure) of the docking station is correctly screwed in its seat

b. Place the WiTP wireless in its housing (docking station) on the Control Unit

c. Make sure that the Teach Pendant STOP pushbutton is released, rotating it clockwise

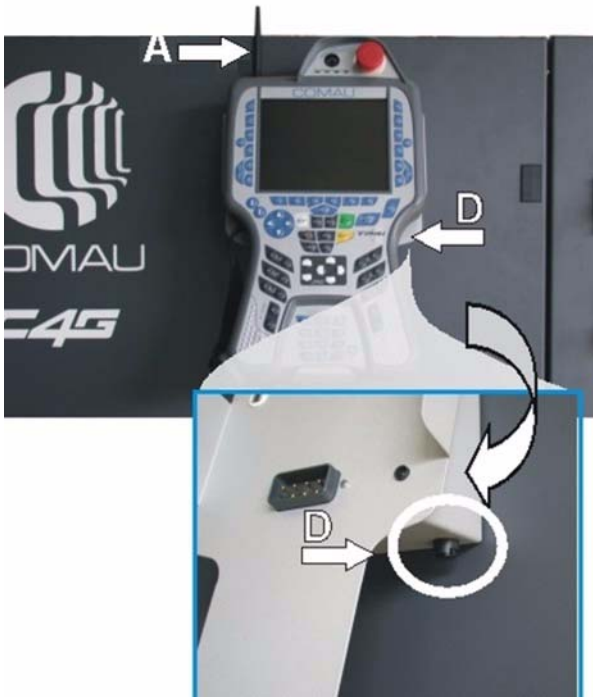
d. Switch on the WiTP wireless pressing pushbuttons AUX B+ and AUX B- simultaneously for a few seconds (see par. Keys to switch on the WiTP wireless on page 1-31 of the Basic Training Course **First Day**)

e. Wait for the teach pendant functioning tests to finish

f. The screen shown on here on the left is then displayed.

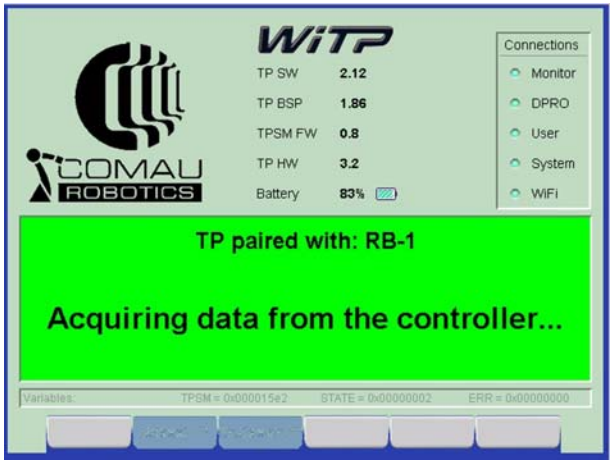
g. Press the “Pairing Request” pushbutton (D) in the lower section of the housing. See the fig. on the left.

h. Wait for messages and instructions on the WiTP wireless display and make the indicated tests; the message “Pairing in progress” appears and the LEDs are updated at the end of each successfully executed connection (WiFi, System, etc.).



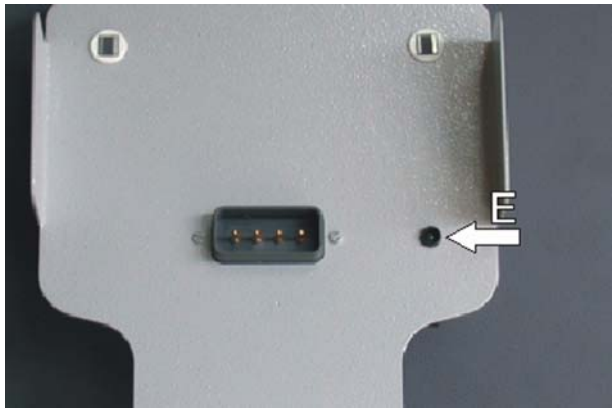
i. The procedure asks the user to press the Teach Pendant STOP pushbutton, then to reset it by rotating it clockwise (see **the two figures here below).**

j. If the result is correct, a message is shown informing that the connection has taken place. At the end of the data acquisition by the Controller, the pairing page closes automatically. Pairing has been completed.



If the user left the timeout to expire before pressing or releasing the STOP pushbutton, the pairing procedure is interrupted. To execute it again, start from step a.

k. The green LED alight on the docking station (E) indicates that pairing has taken place. See the figure here beside, on the right.



3.2 Unpairing between WiTP Teach Pendant and Control Unit

This procedure, called ‘unpairing’, is necessary when the user wishes to terminate the pairing between the C4G Control Unit and the WiTP wireless Teach Pendant. This may be, for example, when it is wished to shut down the system, and pair the WiTP wireless with another Control Unit, replace it, etc.

- **Use the WiTP wireless currently paired with the Control Unit.**
- **The UnPairing procedure can also be carried out with the system in automatic mode.**

a. Place the WiTP wireless in its housing (docking station).

b. Press the “Unpairing request” pushbutton (D), in the lower section of the docking station (see the first figure in the Pairing procedure between WiTP wireless Teach Pendant and C4G Control Unit section).

c. A screen page is shown that updates the user on the current state of the unpairing procedure. Wait until the procedure is completed. See the figure here on the right.

d. The LEDs are updated at the end of every disconnection (WiFi, System, etc.) that has been executed successfully.

e. When the pairing procedure page is shown again, the unpairing operation has been completed (see the figure here beside, on the left)

f. Remove the teach pendant from the support.

g. If it is NOT wished to use the teach pendant on another Control Unit, it should be shut down using softkey Shutdown TP (F3)

WARNING! Do not leave the WiTP wireless near any other Control Unit: the STOP pushbutton is not active, but could confuse an operator who does not know about this limitation.

3.3 Emergency Unpairing

This procedure is necessary in any situation where the WiTP wireless is no longer “sensed” by the C4G Control Unit (the teach pendant may be off, too far away to b recovered, lost, damaged, etc.). When the Control Unit is unable to "find" the Teach Pendant paired to it, the green LED flashes (E). See the last figure of the Pairing procedure.

This procedure is not necessary if the distance of the WiTP wireless is only temporary: when it returns into the range of action, the connection with the Control Unit reactivates automatically.
Obviously it is not necessary to run the emergency unpairing procedure when the green LED is OFF: this means there is no pairing.

The Control Unit has to remain on, during the entire procedure.

a. DO NOT insert the WiTP wireless in the docking station

b. Set the system in programming mode (modal selector switch set on T1)

c. Open the C4G cabinet door

d. Temporarily remove connector X111 from board FIA3 (located on the left wall inside the cabinet). See the figure on the right.

e. Press the pairing pushbutton; see the first figure of the Pairing procedure between WiTP wireless Teach Pendant and C4G Control Unit.

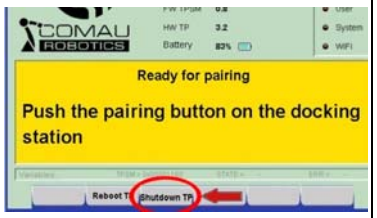
f. Wait for the procedure to finish, indicated by the green LED switching off on the docking station. See the last figure of the Pairing procedure between WiTP wireless Teach Pendant and C4G Control Unit

g. Refit connector X111 on board FIA3. See the figure on the right.

h. The Teach Pendant is ready to execute the pairing procedure.

4 System Restart with WiTP wireless

In case of some system restart Events, the WiTP wireless has to be handled in a different way (summarized in the following table):

Event	Behaviour	Errors generation	Maintains pairing status
Restart with the 4 keys (see par. Keys to switch on the WiTP wireless on page 1-31 of the Basic Training Course **First Day**)	TP restarts, but C4G doesn't	If the WiTP wireless is paired to a Controller, a connection with TP error is generated	Yes: after the restart, the teach pendant reconnects to the C4G it was paired to
Restart cold command on TP or WinC4G user interface	C4G and TP restart	No connection error is generated	Yes: after the restart, the teach pendant reconnects to the C4G it was paired to
 Restart with rear reset pushbutton	TP restarts, but C4G no	If the WiTP wireless is paired to a Controller, a connection with TP error is generated	NO: THE TEACH PENDANT DOES NOT RECONNECT. IF BEFORE THE RESET IT WAS PAIRED, AN Emergency Unpairing OF THE C4G IS NECESSARY
Shutdown from softkey (Shutdown pushbutton) and power on again 	The TP shuts down and switches on again as required	*[Since this command is only available if the teach pendant is NOT paired, there is no connection error]*	*[Since this command is only available if the teach pendant is NOT paired, there is no impact on the pairing status]*
Shutdown due to discharged battery (when the charge drops below 8%, the system asks the user to place the WiTP in the docking station)	The TP informs the user with sufficient notification. If not returned onto the docking station in useful time, it switches off	If the WiTP wireless is paired to a Controller, a connection with TP error is generated	NO: THE TEACH PENDANT DOES NOT RECONNECT. IF BEFORE THE RESET IT WAS PAIRED, AN Emergency Unpairing OF THE C4G IS NECESSARY

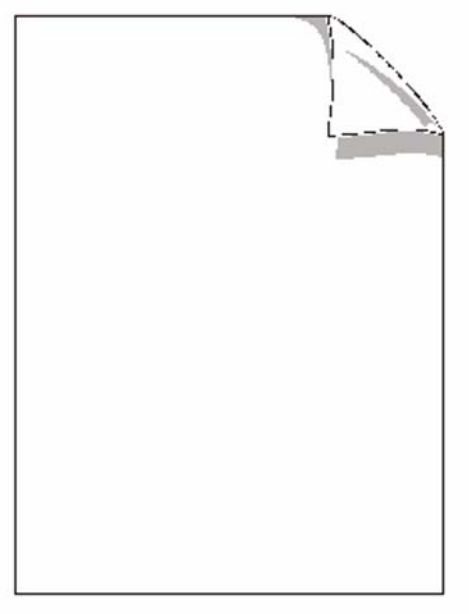
5 Hints for using the WiTP wireless

- To use the WiTP teach pendant paired to the C4G Control Unit, check the Control Unit ID. displayed on the teach pendant Home Page.
- Do not move with the WiTP out of the range of action, to avoid interrupting the communication and the consequent emergency stop of the cell, or system error. In any case, communication is automatically resumed when the teach pendant returns inside the range of action.
- When in use, check the battery charge that remains (eighth field of the Status bar); if the indicator is red, recharge, placing the teach pendant in its docking station. To have 6 hours operation autonomy, it has to be left to recharge in its housing for at least 2 hours, with the Control Unit on.
- when not in use, it is strongly recommended to leave the Teach Pendant in the docking station of the Controller it is paired with.
- If it is necessary to replace it or move it to a different Control Unit, FIRST OF ALL carry out the Unpairing between WiTP Teach Pendant and Control Unit.
- If the WiTP is placed on the docking station of another C4G Control Unit, or the battery is allowed to run down completely, the pairing is automatically removed. To recover it, after removing the cause, it is necessary to run the Emergency Unpairing procedure.
- If communication between Control Unit and WiTP fails, and the robot has the motors running, the system generates an alarm

  `28936 – 10 Emergency stop: WiTP disconnected`

  and the motors shut down.

For further details see the following manuals:
- **C4G Control Unit - Technical Specifications;**
- **C4G Control Unit - Transport and Installation;**
- **C4G Control Unit - Maintenance.**



Basic Training Course - C4G USE AND PROGRAMMING - Fifth Day

1 Basic information

Application: co-ordinated set of both **Hardware** (different devices which co-operate) and **Software** (user interface in mutual relation with the hardware, dedicated to the application, but open and flexible) suitable to provide a specified performance.

There are several Comau standard application types. The most commonly used are as follows:

– SmartHand System - Handling Application (Hardware manual: depending on the chosen robot - Software manual: Handling System),

– SmartSpot System - Spot Welding Application (Hardware manual: depending on the chosen robot and welding type - Software manual: Spot Welding System),

– SmartArc System - Wire Welding Application (Hardware manual: depending on the chosen robot - Software manual: Wire Welding System),

– SmartIP System - Interpress Handling Application (Hardware manual: Interpress Handling System CAN Bus - Software manual: Interpresses Handling System),

– SmartGlue System - Sealant Application System (Hardware manual: Sealant Application System - Software manual: Sealant Application System),

– SmartStud System - Stud Welding Application (Hardware manual: Stud Welding System - Software manual: Stud Welding System),

– SmartLaser System - Laser Welding Application (Hardware manual: SmartLaser™ - Software manual: SmartLaser™),

– ToolChange System - Automatic Tool Change Application, for handling and welding applications (Hardware manual: Automatic Tool Changer).

Starting from the chosen application, the user is allowed to

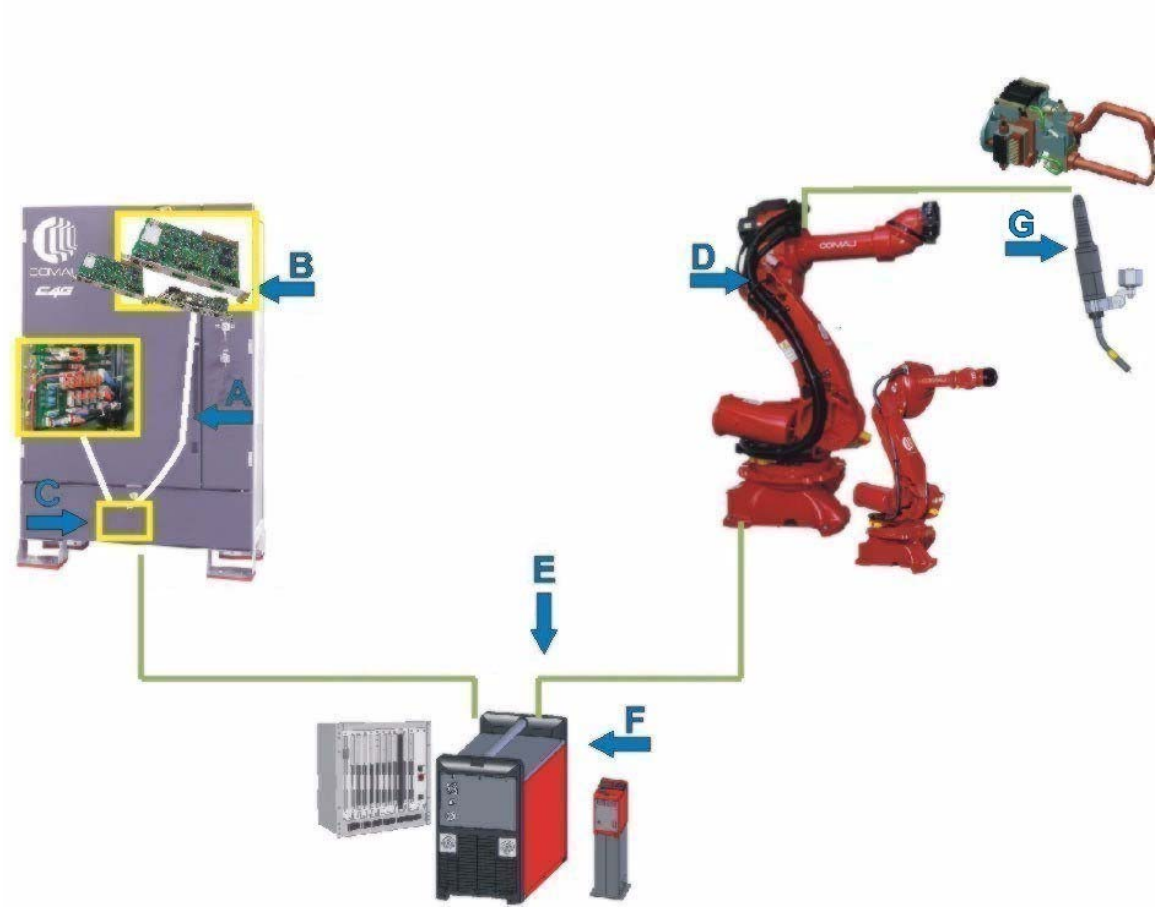
– directly use it, since it is extremely flexible, thus suitable to satisfy to various requirements;

– add Comau options;

– self-customize the application.

It is suggested to read the specific application manuals, to get further information.

1.1 Hardware



The following components are common to any application:

- **C4G Control Unit**: common to all the applications.
- (**B**) **Fieldbus boards** to be installed inside the Control Unit: DeviceNet, Profibus-DP or Interbus-S. The CANbus board is already present inside C4G Control Unit.
- (**A**) **Fieldbus cable and power supplies, cable installed inside the Control Unit, to supply** 24Vcc from X137/FIA connector, and signals from the Fieldbus boards.
- (**C**) **X93 Multibus Connector** on the cabinet base.
- (**E**) **Multibus Cable, external to the Control Unit**, allowing to connect the Control Unit to external devices having the same bus and a suitable connector:
  - (**F**) tool check **device** (if needed)
  - (**G**) **tool mounted on the robot** - e.g. welding gun
  - additional devices placed close to the robot.
- Depending on the Application type, the following components might be needed (**D**) **Outfitting -** tubes and cables towards the robot, to the mounted tool.
- **Robot** of N family (NS, NM, NH, NJ, Six), to be chosen depending on the Application type and on the tool type.

1.2 Software

The basic functions of an Application are as follows:

- User Interface on the Teach Pendant
- Integrated developping environment
- Technological Instructions Set
- Process dedicated Keys
- Alarms handling
- I/O signals configuration.

User Interface on the Teach Pendant

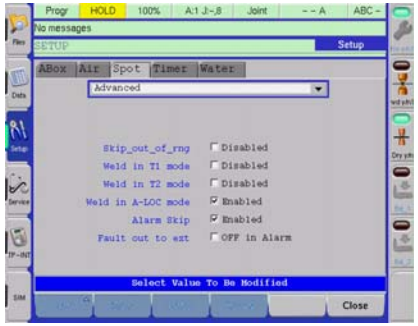
Appl Page

Displays the Application status

I/O Page

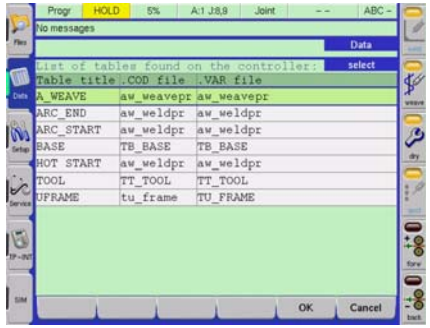
Displays the I/Os status

Setup page



Allows to setup the application parameters

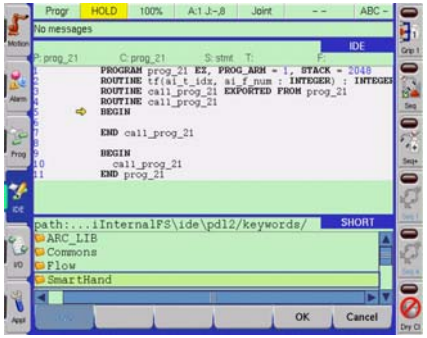
Data Page



Handles the application tables

Integrated developping environment

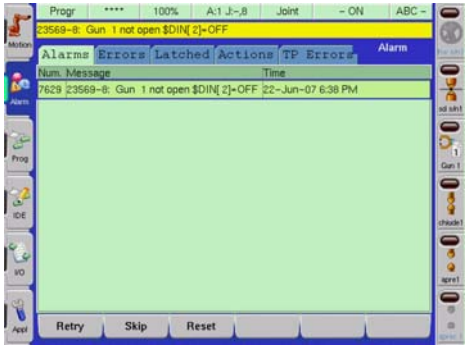
IDE Page



Allows to write, modify and tune PDL2 programs

Alarms handling

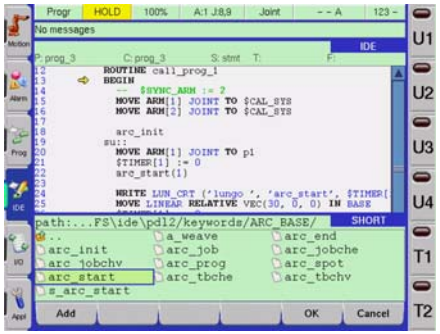
Alarm Page



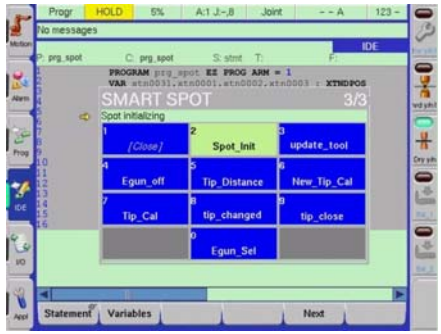
Allows to handle both alarm and information messages.

Technological Instructions Set

Inserting the application technological instructions



standard method



Numeric keypad

Process dedicated Keys

Allows to use the TP Right Menu keys for special functionalities, dedicated to each application (see the figure on the right).

I/O signals configuration.

Allows to configure the I/O ports used by the Application (see Programs for Fieldbus and I/O configuration).

1.3 Installation Procedure

When the System is delivered, the Application software is already properly installed. The need of installing an Application again could occur in one of the following situations:

– **the Application software is already installed but it is not possible to activate it** - this is the case in which, by mistake, one or more files belonging to the application software, have been deleted from the User Directory;

– **application software upgrading (installation of a newer software version, with respect to the currently installed on C4G)** - this is the case in which an upgrading is wished of the application software version, keeping the current configuration.

The following operations are necessary to install the application:

a. Activate the SETUP page on the Programmaing Terminal

b. Select the Install icon and confirm with ENTER.

c. Press ENTER again, then down arrow to select the device containing the SmartHand application to be installed.

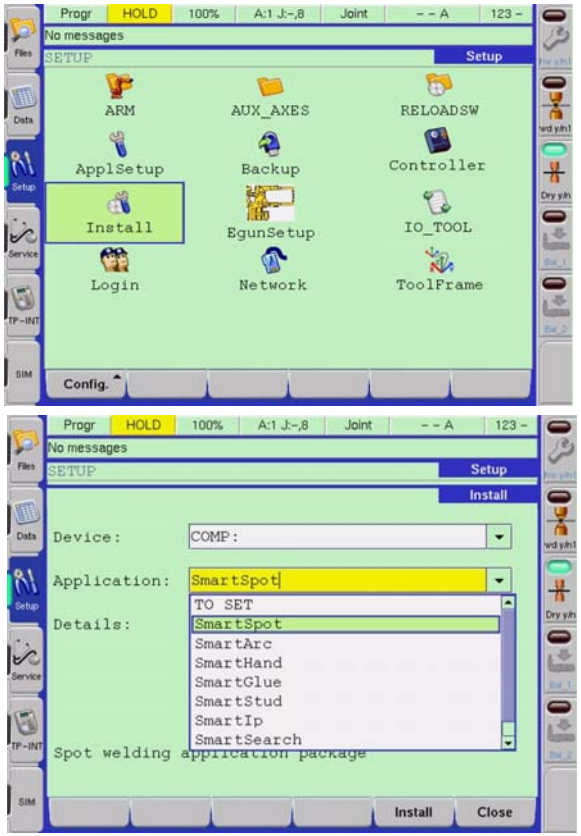
d. Move the cursor to the next field and select the being installed application (ENTER to open the menu, up/down arrow key to select). In the low right screenshot, an example is shown of the SmartSpot System - Spot Welding Application software installation.

e. Press ENTER to confirm

f. Press the key for the Install command (F5) to carry out the installation.

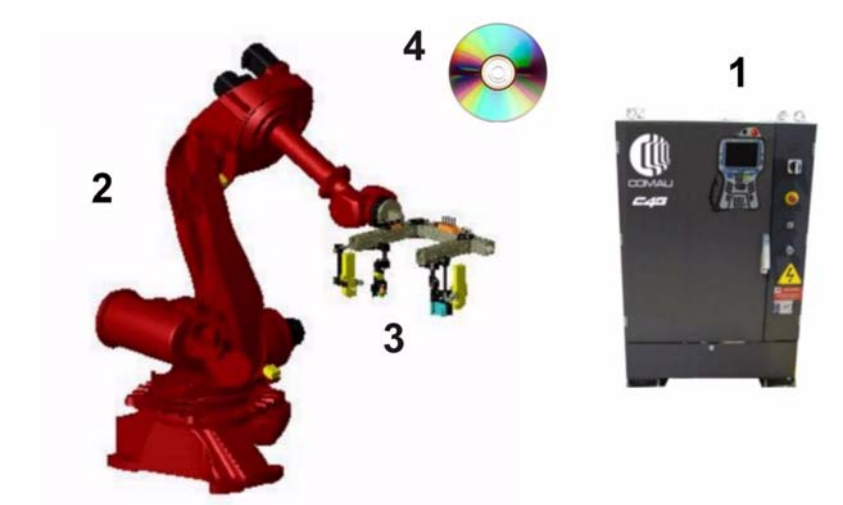
g. In case of an error the installation is stopped, the error is indicated with a message and the C4G red alarm lamp lights up. The system remains in the condition until the RESET key is pressed.

h. After the installation has been completed, re-start the system.



The following Applications are shortly described in the next paragraphs:
– SmartHand System - Handling Application,
– SmartSpot System - Spot Welding Application,
– SmartArc System - Wire Welding Application.

2 SmartHand System - Handling Application



SmartHand is the simplest Comau standard applications, thus a basis for all other Comau applications.

The Handling system makes it possible to use a gripper or similar fixture with electro-pneumatic control coupled to a robot.
It is an application composed by:

1 - C4G Robot Controller Unit

2 - Robot SMART NHx, NS, NM, NJ, Six, NX1

3 - External tool to execute the technological process (e.g. a gripper)

4 - **SmartHand** application software **-** Handling System**.**

Furthermore, SmartHand System also include tubes and cables towards the tool mounted on the robot and which performs the technological process (referred to as **Outfitting**) and some possible options (e.g. tool changer, etc.).

2.1 Hardware Components - Configurations available

For the simplest solution, I/Os on DSA, the following components are required:

– C4G Control Unit
– either NHx or NS robot
– Robot Outfitting
– either I/O connection available on robot board,
– or I/O connector kit (optional)

For a more complex solution, Fieldbus, the following components are required:

– C4G Control Unit
– either NHx or NS robot
– Fieldbus boards
  • Device Net, or
  • Profibus-DP, or
  • Interbus-S
– Bus cable and power supply, provided together with the Fieldbus board
– Multibus cable
– On Robot Outfitting
– External Outfitting (bazooka), or
– Handling connection kit (optional).

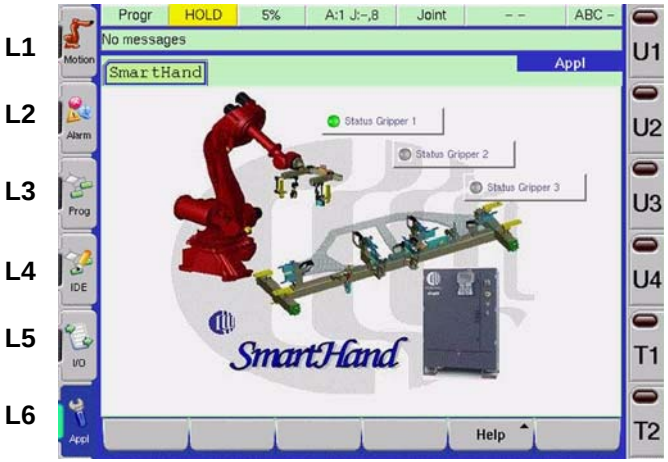
NOTE THAT: if the chosen robot is NOT NH4, the Empty tube kit is also required. It is an empty tube that can be used as a protection sheath for the tubes and/or cables installed by the integrator.

2.2 Software Components - Handling System

Il software applicativo SmartHand, installato sull'Unità di Controllo C4G, è necessario per gestire in modo semplice ed omogeneo un Sistema di Manipolazione.

– User Interface on TP
– Technological Instructions Set
– Process control keys (right menu).

2.2.1 User Interface on TP



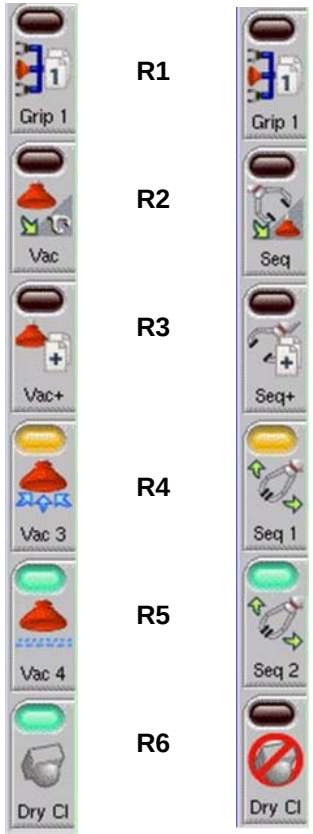
It is the Home Page of SmartHand Application software: it displays the status of the grippers used by the handling system. To activate it, press **Appl** (**L6**) key, in the Left Menu.

2.2.2 Technological Instructions Set

The user is provided with a set of Technological instructions dedicated to the Handling process, especially for the Gripper tool (see par. 5.3 Technological instructions on page 5-3 in Handling System manual); they can handle the clamps opening and closing, the suction device activating, stopping and blowing, the part presence or lack, the air pressure controlling.

2.2.3 Process control keys (right menu)

The right menu includes keys and “software leds” for handling the process. Please refer to the two examples on the right.



- R1 key selects the desired gripper.
- R2 selects the type of component to be displayed/controlled (Vac/Seq). Each time it is pressed the icons update according to the type of components configured on the gripper indicated by R1.
- R3 key increases the number of the component currently selected by R2 key.
- R4 and R5 keys execute the commands which are indicated by their icons:
  - **blowing/aspiration** command when Vac is selected,
  - **open/close** command when Seq is selected.
- R6 key handles the **DRY CYCLE** (cycle with no parts) command.

Depending on the modal selector switch position, each key may be enabled or not.

The “software leds” colors states for:
- grey - unknown or not influent state, corresponding to the component represented by the icon;
- yellow - command already sent; waiting for execution completed;
- green - command executed.

The prohibition signal, on the DRY CYCLE icon, indicates that the dry cycle execution is disabled.

3 SmartSpot System - Spot Welding Application

The Spot Welding System performs the welding applying 50 Hz or 1000 Hz using electric gun or pneumatic gun coupled to a Robot.

It is composed by:
1 - C4G Control Unit, containing a Fieldbus Board
2 - SMART NHx, NJx Robot
3 - Pneumatic or Electric Welding Gun
4 - **SmartSpot** application software **-** Spot Welding System
5 - Welding box and Media Panel

Furthermore, SmartSpot System also include water and air tubes, cables towards the welding gun mounted on the robot and which performs the technological process (referred to as **Outfitting**) and some possible **Equipment**s to control devices mounted on the robot and/or other devices installed near the robot, as well as other possible **option**s (e.g. tip dress, tip change, etc.).

3.1 Hardware Components - Configurations available

Depending on using either the penumatic gun or the electric gun, on robot board or stationary, the following configurations may occur:

- Pneumatic weld gun - stationary
- Electric weld gun - stationary
- Pneumatic weld gun - on board Robot
- Electric weld gun - on board Robot

Pneumatic weld gun - stationary

- C4G Control Unit
- NH, NJ Robot
- Fieldbus board
  - Device Net, or
  - Profibus-DP, or
  - Interbus-S cable
- Cables (bus, power supply, internal cable, Multibus, power cables between welding box and robot)
- Welding Box and Media Panel
- Air/water tubes kit between Media Panel and robot
- Options (tip dresser unit, tip changer unit, etc.) and corresponding cables.

Electric weld gun - stationary

- C4G Control Unit
- NH, NJ Robot
- Fieldbus board
  - Device Net, or
  - Profibus-DP, or
  - Interbus-S cable
- Cables (bus, power supply, internal cable, Multibus, power cables between welding box and robot)
- Welding Box and Media Panel
- Air/water tubes kit between Media Panel and robot
- Additional axis kit and connecting cables between additional axis and robot
- Options (tip dresser unit, tip changer unit, etc.) and corresponding cables.

Pneumatic weld gun - on board Robot

- C4G Control Unit
- NH, NJ Robot
- Fieldbus board
  - Device Net, or
  - Profibus-DP, or
  - Interbus-S cable
- Cables (bus, power supply, internal cable, Multibus, power cables between welding box and robot)
- Welding Box and Media Panel
- Air/water tubes kit between Media Panel and robot
- Spot Welding Outfitting on board Robot, plus external section
- Spot Pneumatic connection kit (as alternative to bazooka Outfitting).
- Options (tip dresser unit, tip changer unit, etc.) and corresponding cables.

Electric weld gun - on board Robot

- C4G Control Unit
- NH, NJ Robot
- Fieldbus board
  - Device Net, or
  - Profibus-DP, or
  - Interbus-S cable
- Cables (bus, power supply, internal cable, Multibus, power cables between welding box and robot)
- Welding Box and Media Panel
- Air/water tubes kit between Media Panel and robot
- Spot Welding Outfitting on board Robot, plus external section
- Additional axis kit and connecting cables between additional axis and robot
- Options (tip dresser unit, tip changer unit, etc.) and corresponding cables.

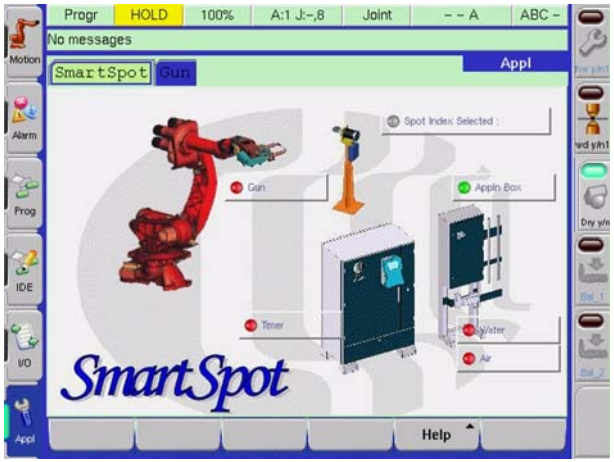
3.2 Software Components - Spot Welding System

SmartSpot application software, installed on C4G Control Unit, is required to properly handle a Spot Welding System.

- User Interface on TP
- Technological Instructions Set
- Process control keys (menu on the right).

3.2.1 User Interface on TP

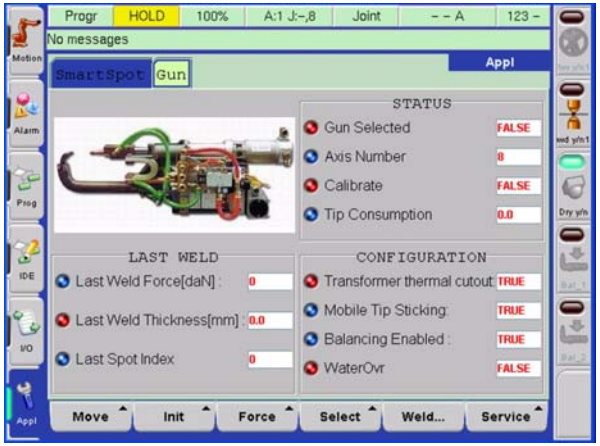
L1 L2 L3 L4 L5 L6



R1 R2 R3 R4 R5 R6

It is the Home Page of the SmatSpot application software (see figure on the left): it displays the state of all the Spot Welding System devices. Press the **Appl** (**L6**)key in the Left Menu on the Teach Pendant.

If the system handles more than one servo gun, a Gun page will be available for each gun (for ex. Gun1, Gun2, etc.). If only one gun is available, the page name will simply be Gun. See the figure here below.



3.2.2 Technological Instructions Set

The user is provided with a set of Technological Instructions specifically dedicated to the Spot Welding process (see par. 5.3 Technological Instructions on page 5-4 in Spot Welding System manual); they allow handling the welding process, communicating with the welding timer, moving and controlling the welding gun (both penumatic and electric), handling tip changing, etc.

3.2.3 Process control keys (menu on the right)

The right menu includes keys and “software leds” for handling the process. The icon and led colours highlight the status of the corresponding softkey and function for the user. The general rule to understad the colours is the following one:

– icon colour:
  • grey icon - softkey pressing disabled
  • coloured icon - softkey pressing enabled

– led colour:
  • black led - associated function OFF
  • green led - associated function ON
  • yellow led - command in progress or inconsistency between command and associated function status

Application inclusion/not inclusion key - The R1 key allows not including or including the application (specified by the index on the icon). Application not included is a status where the application works in deteriorated mode, as if the hardware devices associated to the application itself (timer, application box, media panel) did not exist; therefore, the application DOES NOT carry out any (command and/or checking) activity on them, EXCEPT on the ones that could damage the operator or equipment safety (welding remote switch, status, sensors, gun arms position).

R1

WELD YES/WELD NO key (R2 key) - The WELD YES/NO modes can be selected when application included, and differ for the actual execution of the welding cycle, featuring or not a current flow through the gun electrodes.

R2

PLEASE NOTE - The enabling of the R1 and R2 keys can be configured by the user as a function of the intended application.
By default, the R1 and R2 keys are enabled only when the Control Unit mode selector is in the AUTO (Local Automatic) position. It is possible to setup their enabling/disabling functionality through the TP keys menu, SPOT subpage, in the SETUP Page.

Electric gun motion keys - They work as the gun keys but they are related to the pneumatic balancing cylinder of the electric gun. It deals with toggle-like keys, (status changes at each pressing); each one relates to a balancing solenoid valve, configured for the corresponding gun.

(R3) - It refers to the DRY-CYCLE (cycle featuring no item), applied in case of electric gun. When the dry-cycle is on and the application package is included, the welding cycle is carried out normally (both by WELD YES and WELD NO), but the plate thickness value, (coming from timer or table), that will be used by the electric gun during the force exerting phase, is set to zero.

Pneumatic gun motion keys -

gun closed gun open gun fully open

It deals with the R3, R4 and R5 keys. They carry out the manual motion of the pneumatic welding gun.

By pressing one of those keys, under enabled condition (T1 + Enabling Device), the command to move the gun to the corresponding position will be carried out.

PLEASE NOTE - In case of configurations featuring more guns, controlled by the application package, in the menu on the right an additional icon will be displayed, which enables to select the number of the gun, on which the right keys shall act.

4 SmartArc System - Wire Welding Application

SmartArc is the Wire Welding Application System.
A Wire Welding system is subject to different configurations according to the part to be welded, the type of joint and the welding process to be used.

The basic system includes:

1 - C4G Robot Controller Unit
2 - SMART NS, Six **Robot**
3 - Welder
4 -Wire puller on board robot
5 - Torch
6 - Safety flange
7 - Torch cleaning-lubrication device
8 - SmartArc application software **- Wire Welding System.**

Furthermore, SmartArc System also include the on board Robot Outfitting (power cable and signals), all connecting cables and the possible options (e.g. Joint Search wire Sensor, Seam track, etc.).

4.1 Hardware Components - Configurations available

The following configurations of Wire Welding System Application are available depending on:

– used welding process

– communication mode between C4G Control Unit and generator

– cooling type (either water or gas).

4.2 Software Components - Wire Welding System

The basic functions, always available in a Wire Welding system, are the following:

– User Interface on TP

– Technological Instructions Set

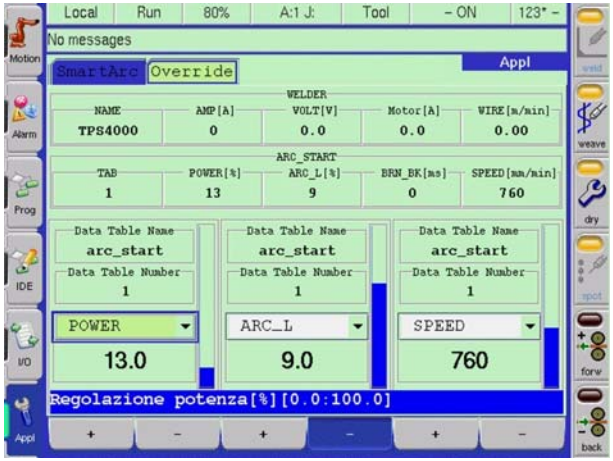
– Process control keys (right menu).

4.2.1 User Interface on TP

L1
L2
L3
L4
L5
L6

It is the Home page of SmartArc application software: it shows the state of the Basic System components. To activate it, just press **Appl** (**L6**) key in the Left Menu on the Teach Pendant.

An **Override** page is also available (as shown in the figure here below); it includes some information related to the welding process and allows the user to **real-time modify** some parameters related to process, weaving and robot speed.

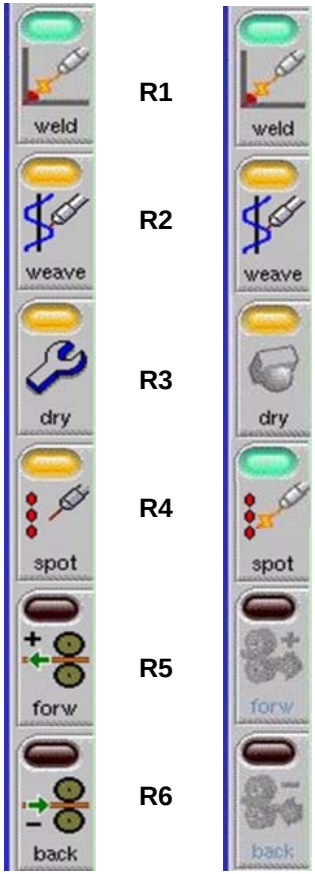


4.2.2 Technological Instructions Set

The user is provided with a set of technological Instructions specifically dedicated to Wire Welding (see par. 5.2 Technological instructions on page 5-1 in Wire Welding System manual); they handle process functions initialization, selection of program, welding and/or weaving job, arc off mode, welding and/or weaving tables, etc.

4.2.3 Process control keys (right menu)

The right menu includes keys and “software leds” for handling the Wire Welding process. Please refer to the two examples on the right.



- **R1** key activates/deactivates the weld function - WELD ON/OFF.
- **R2** key activates/deactivates the weave function - WEAVE ON/OFF.
- **R3** enables/disables the dry run function (cycle with NO execution of technological instructions, even if existing in the working program).
- **R4** key displays the state of the intermittent arc, enabled by ARC_SPOT (optional) instruction. When the option has not been purchased, such a display is not active.
- **R5** and **R6** keys handle wire feed and wire back functions, respectively.

Depending on the modal selector switch position, each key in the right menu may be enabled or not.

The “software leds” colors stand for:
- grey - the procedure is not in progress;
- giallo - function enabled but not currently in progress;
- verde - function in progress.

As far as the icons visibility, the meaning is as follows:
- not visible icon = the functionality is disabled (pressing the key does not take any effect)
- visible icon = the functionality is enabled (pressing the key causes to activate/deactivate it)

5 Collision Detection (optional feature)

- Basic Concepts
- Activation/deactivation of Collision Detection function
- Activation/deactivation of the Compliance
- Collision Detection sensitivity type
- Reliability
- Use of the Collision Detection functionality
- Managing "collision detected" event (program structure sample).

5.1 Basic Concepts

The Collision Detection algorithm (optional feature) allows the system to stop the robot arm motion, as soon as any noise force on joints takes effect.
Such a capability can be enabled and disabled by the user, by means of a predefined variable, using either a Program statement or a system Command

The Collision Detection functionality has NOT been designed in order to protect the personnel, but to limit any damage to the robot mechanical parts and therefore to its equipments!

When a collision condition is detected (error message 62513SAX: collision detected), the system may enter into a particular phase during which the robot gets compliant and the emergency braking takes place
The axes compliance has been used in order to be able to absorb part of the collision energy (or part of the over-traction in case of stuck tips) and to minimize any possible damage, this compliance can be activated by the user. Carefully assess its use in the application.
Depending on the needs, upon a collision detection, it is possible to:

- generate a DRIVE OFF, or
- put the machine in HOLD state.

It is also possible to trigger the Collision Detection event only (system event no.197). The robot DOES NOT STOP, but continues the programmed motion: it is the responsibility of the user to appropriately deal with the event that has occurred, cancelling the current motion, that would continue towards the obstruction, and programming the new trajectory. To do that, see the sample program structure in par. 5.7 Managing "collision detected" event on page 5-23.

5.2 Activation/deactivation of Collision Detection function

In order to enable the Collision Detection functionality, it is enough to set $CRNT_DATA[num_arm].COLL_ENBL predefined variable to TRUE, by either a Program Statement or a System Command:

```
$CRNT_DATA[num_arm].COLL_ENBL := TRUE
```

By default, it is not reset by the system upon each DRIVE OFF command; it continues to be working until the user consciously decides to disable it, by setting the mentioned above flag to FALSE.
It is also possible to ask for automatically disabling it, setting to 1 bit no.10 of the $ARM_DATA[num_arm].A_ALONG_1D[12] system variable, by means of the BIT_SET Built-In Procedure.

```
BIT_SET($ARM_DATA[num_arm].A_ALONG_1D[12], 10)
```

The bit modification is allowed both at run-time and at configuration level, and it is possible to save it in .C4G file.

5.3 Activation/deactivation of the Compliance

If, upon a collision alarm, the robot stop is wished to be made compliant, the user must enable the compliancy functionality by means of the suitable PDL2 statement:

```
BIT_SET($ARM_DATA[num_arm].A_ALONG_1D[12], 11).
```

It is also possible to disabled it, by means of the following PDL2 statement:

```
BIT_CLEAR($ARM_DATA[num_arm].A_ALONG_1D[12], 11).
```

To avoid that this bit is unintentionally saved in .C4G and unexpectedly enables the compliance service related to the Collision Detection after a restart, the bit itself will be automatically reset by the system during the start-up.

5.4 Collision Detection sensitivity type

A detailed description follows of predefined variables which are involved in the Collision Detection functionality. They are:

– $COLL_TYPE

– $ARM_SENSITIVITY (sensitivity threshold of the axes)

– $COLL_SOFT_PER (axes compliance thresholds).

5.5 Reliability

WARNING! In order to make the Collision Detection algorithm to operate in a safe and reliable way, it is ABSOLUTELY REQUIRED that the user properly identifies the payload.
The Collision Detection performance strictly depends on a proper declaration of the currently used payload: if the estimated payload is wrong, the Collision Detection could be misevaluated (i.e. false collision during a Program execution); furthermore, while in the compliance phase, with consequent deceleration and stop, the robot arm could have unpredictable behaviours due to either an underestimated or an overestimated payload.

For these reasons, before activating the Collision Detection function, the user is to check that the load used has been defined precisely. To do that, it is needed to use the Payload identification (optional function) procedure.

5.6 Use of the Collision Detection functionality

Since this functionality is complex and delicate, it is strongly suggested to carefully read its full description in the **MOTION PROGRAMMING** manual, par. 15.6.2 Use of the Collision Detection Procedure on page 15-7.

In such a paragraph, all information is supplied to properly setup the Collision Detection predefined Variables,

– To identify the thresholds for that work cycle,

– For Splitting-up the program to make the performance more efficient and effective,

– For Activating the functionality,

– To handle False collisions,
– To setup the Compliance thresholds,
– etc.

Before going on in using the Collision Detection functionality, it is strongly recommended to carefully read such a paragraph!

5.7 Managing "collision detected" event

Simply as an example, a program segment is given that manages the "collision detected" event 197 .

```
PROGRAM colltouch
VAR pnt0006p, pnt0007p, pnt0008p: POSITION
pnt0001p, pnt0002p, pnt0003p, pnt0004p, pnt0005p: POSITION
BEGIN
  CONDITION[1]:
       WHEN EVENT 94 DO
       UNLOCK -- reset for
       RESUME -- next restart
         .
  ENDCONDITION
  CONDITION[2]NODISABLE:
       WHEN EVENT 197 DO
         LOCK              -- machine locked
         CANCEL CURRENT    -- cancel current motion
         .
  ENDCONDITION
     ...
  ENABLE CONDITION[1]
  ENABLE CONDITION[2]
     ...
  $COLL_TYPE:=COLL_USER1
```

```
  $COLL_EFFECT:=2
  $CRNT_DATA[1].COLL_ENBL:=TRUE

  CYCLE
    MOVE TO ...
          .
END colltouch
```

6 Payload identification (optional function)

The new generation robots (N families) are fitted with a complete dynamic model on all six axes.
Because of the model's sensitivity to the declared payload parameters, it is necessary that the user checks in every situation the correct definition of the payload used. An imprecision in the declaration of the payload characteristics has a negative influence on the robot performance.

Warning: for safety reasons it is very important that the machine is not used before having defined the values for the payload. The omission or a big error in the payload declaration could create potentially hazardous situations for the users and for the fixtures

As the correct declaration of the tool dimensions (variable $TOOL) is necessary to obtain the required precision in execution of movements in the Cartesian environment, in the same way it is equally necessary to correctly define the payload parameters that will have influence on the robot movements in terms of performance optimisation and safeguarding the components.

The determination of the payload characteristics takes place using the specific procedure.

The Payload Identification procedure calculates again the values of the following predefined variables:

– $TOOL_MASS (expressed in kg)

– $TOOL_CENTER (x, y and z)

– $TOOL_INERTIA[1..6] .

Once this has been determined, the parameters will be immediately active and automatically stored in the specific table created by TO_SET (TT_TOOL1.var), to be called up when executing the movement cycles, each time the same payload is used.

6.1 Procedure

The motion programs for the Payload identification are supplied by Comau in PDL2 language and differ depending on the robot model, on the basis of the robot kinematic features. Their use is strongly recommended , unless there are physical impediments in the robot working area.
For detailed information, refer to par. 18.4.7.2 Procedure on page 18-23, in **Motion Programming** manual.

The user is suggested to carefully read such a paragraph, before going on executing the Payload identification procedure!

Basic Training Course - C4G USE AND PROGRAMMING - Appendix 1

– Basic concepts

1 - Introduction to PDL2
1.2 - Data types
1.3 - Routines
1.4 - Sharing Variables and Routines

– PDL2 Statements

2 - Basic PDL2 Statements
2.1 - Flow Control Statements
2.2 - Program Control Statements
2.3 - File Manipulation Statements
2.4 - Motion Statements
2.5 - Bit Manipulation Built-in Routines
2.6 - ASSIGNMENT statement

1 Introduction to PDL2

The current chapter describes just the BASIC concepts about PDL2 Language.
For a full description, the Advanced PDL2 Programming Training Course is available.

1.1 Data items

PDL2 programs can include the following kinds of data items:

– VARIABLES, representing values that can change;

– CONSTANTS, representing values that cannot change;

– LITERALS, actual values.

Variables and constants are defined by an identifier and a data type.

A VARIABLE declaration establishes a variable identifier with two attributes: a name and a data type. Within the program, the variable can be assigned any value of the declared data type. Ex.: VAR ***aaa*** : INTEGER

A CONSTANT declaration establishes a constant identifier with two attributes: a name and an unchanging value. The data type of the CONSTANT is understood by its assigned value, which can be an INTEGER, a REAL, a BOOLEAN, or a STRING. Within the program, the identifier can be used in place of the value. Ex.: CONST ***error1*** = 'file not found'

LITERAL values are actual values used in the program. They can be INTEGER, REAL, or STRING values. Ex.: 123, 98.5, 'abc', etc.

Please refer to the PDL2 program structure example in par. 6 Programs structure overview on page 2-37 of the Basic Training Course - C4G USE AND PROGRAMMING - Second Day, and in par. 3.2 Declarations on page 3-13 of the PDL2 Programming Language Manual, to better understand when and how to declare VARIABLEs and CONSTANTs.

1.2 Data types

The basic PDL2 predefined DATA TYPEs are INTEGER, REAL, BOOLEAN, STRING, ARRAY, POSITION, JOINTPOS.

1.2.1 INTEGER

The INTEGER data type represents whole number values in the range -2147483647 through +2147483647. The following predefined constants represent the maximum and minimum INTEGER values:

– MAXINT;

– MININT.

1.2.2 REAL

The REAL data type represents numeric values that include a decimal point and a fractional part or numbers expressed in scientific notation.

1.2.3 BOOLEAN

The BOOLEAN data type represents the Boolean predefined constants TRUE (ON) and FALSE (OFF).

1.2.4 STRING

The STRING data type represents a series of characters (either ASCII or UNICODE), treated as a single unit of data.

Single quotes mark the beginning and the end of an ASCII string value. Example:

```
'this is an ASCII string value'
```

Double quotes mark the beginning and the end of a UNICODE string value.

```
"this is a UNICODE string value"
```

```
〝程序机构可以进行编辑并示教开发程序〞
```

1.2.5 ARRAY

The ARRAY data type represents an ordered collection of data items, all of the same type.

The programmer can declare that type to be one of the following:

INTEGER, REAL, BOOLEAN, STRING, POSITION, JOINTPOS and some other advanced data types.

The programmer can declare an ARRAY to have one or two dimensions. Individual items in an ARRAY are referenced by index numbers.
Examples:

```
part_1[2] -- indicates the 2nd item of part_1 array
bin[2,3]  -- indicates the item in the 2nd row, 3rd column in the two dimensional bin array
```

1.2.6 POSITION

A POSITION data type represents a CARTESIAN position: it defines

– the LOCATION components (distances, measured in millimeters, along the **x**, **y**, **z** axes of the starting frame of reference),

– the ORIENTATION components (rotation angles, measured in degrees, called **Euler angles**), and

– the robot CONFIGURATION component (for final points of MOVE statements only; it represents a unique **set of the robot joint angles** that bring the TCP on that position).

1.2.7 JOINTPOS

The JOINTPOS data type represents the actual arm joint positions, in degrees. One real component corresponds to each joint of the arm.
Individual components of a JOINTPOS, like ARRAY components, are referenced by index numbers.
Example:

```
PROGRAM jnttest
VAR
  real_var : REAL
  jointpos_var : JOINTPOS
BEGIN
  real_var := jointpos_var[5] -- assigns to a REAL variable the value of the 5th joint
  jointpos_var[3] := real_exp -- assigns to the 3rd joint the value of a REAL expression
END jnttest
```

1.3 Routines

A ROUTINE is structured like a program, although it usually performs a single task.
Routines are useful to shorten and simplify the main program. Tasks that are repeated throughout a program or are common to different programs can be isolated in individual routines.

A program can call more than one routine and can call the same routine more than once. When a program calls a routine, program control is transferred to the routine and the statements in the routine are executed. After the routine has been executed, control is returned to the point from where the routine was called. Routines can be called from anywhere within the executable section of a program or routine.

1.4 Sharing Variables and Routines

To share Variables and/or Routines among Programs, use one of the following methods:

– **EXPORTED FROM** clause - example:

```
PROGRAM prog1
VAR aaa: INTEGER EXPORTED FROM prog1
```

declares that Variable ***aaa*** can be accessed by other Programs

```
PROGRAM prog2
VAR aaa: INTEGER EXPORTED FROM prog1
```

declares that Variable ***aaa*** will be used by Program ***prog2*** and has been declared in Program ***prog1***. It is NEEDED to declare it in ***prog2*** again.

– **IMPORT** statement and **GLOBAL** attribute - example:

```
PROGRAM prog3
VAR bbb, ccc, ddd : REAL GLOBAL
ROUTINE rout1 GLOBAL
```

specifies that Variables ***bbb***, ***ccc***, ***ddd*** and Routine ***rout1*** belonging to ***prog3***, can be accessed by other Programs

```
PROGRAM prog4
IMPORT 'prog3'
```

allows to use any Variables/Routines declared as GLOBAL within ***prog3***, so ***bbb***, ***ccc***, ***ddd*** Variables and ***rout1*** Routine can be accessed by ***prog4***. It is NOT NEEDED to declare them again.

For further information, please refer to par. 3.2.4 Shared types, variables and routines on page 3-18 - PDL2 Programming Language Manual.

2 Basic PDL2 Statements

Basically, PDL2 language provides:

- Flow Control Statements
- Program Control Statements
- File Manipulation Statements
- Motion Statements
- Bit Manipulation Built-in Routines
- ASSIGNMENT statement.

2.1 Flow Control Statements

Statements that control the execution flow within a program are called Flow Control Statements.

The basic PDL2 Execution Control statements are:

– IF Statement

– GOTO Statement

2.1.1 IF Statement

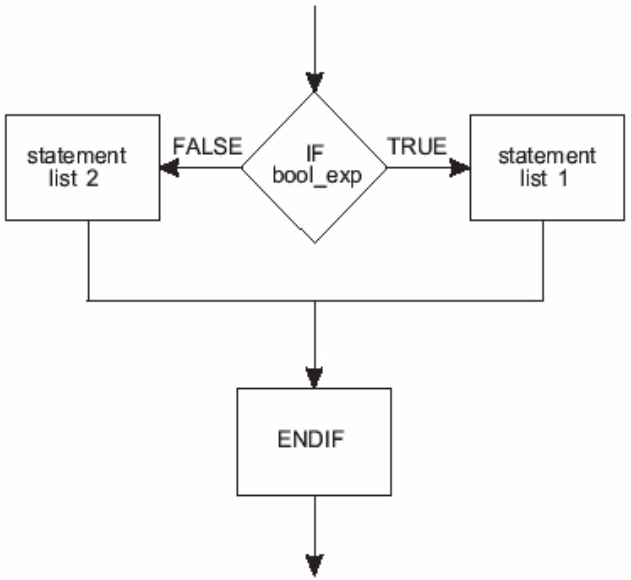
The IF statement allows a program to choose between two possible courses of action, based on the result of a BOOLEAN expression.

If the expression is TRUE, the statements following the IF clause are executed. Program control is then transferred to the statement following the ENDIF.

If the expression is FALSE, the statements following the IF clause are skipped and program control is transferred to the statement following the ENDIF.

An optional ELSE clause, placed between the last statement of the IF clause and the ENDIF, can be used to execute some statements if the BOOLEAN expression is FALSE.

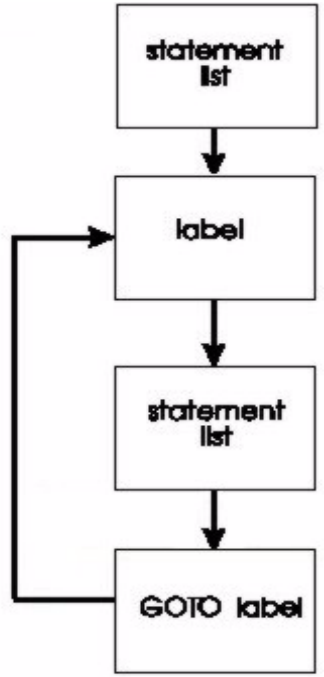
```
IF bool_exp THEN
   <statement_list_1>...
<ELSE
   <statement_list_2>...
ENDIF
```



2.1.2 GOTO Statement

The GOTO statement causes an unconditional branch. Unconditional branching permits direct transfer of control from one part of the program to another without having to meet any conditions.

The GOTO statement transfers the program control to the place in the program specified by a Statement LABEL.



Example:

```
PROGRAM prog_1
BEGIN
  ...
here:: WRITE ('This is where the GOTO transfers to')
  <statement_list>...
  GOTO here
  ...
END prog_1
```

NOTE THAT the Statement LABEL must be within the same ROUTINE or PROGRAM body, otherwise an error occurs.

2.1.2.1 Statement LABEL

The statement LABEL is followed by two consecutive colons (::). Executable statements may follow on the same line, or any line after the label.

See the above example (par. 2.1.2 GOTO Statement on page 6-8) to better understand how to use LABELs.

2.2 Program Control Statements

Statements that control execution of entire programs are called Program Control Statements.

The basic PDL2 Program Control statements are:

– WAIT FOR Statement
– DELAY Statement
– HOLD Statement
– CYCLE and EXIT CYCLE Statements
– RETURN Statement.

2.2.1 WAIT FOR Statement

The WAIT FOR statement causes execution of the program issuing it to be suspended until the specified condition is satisfied. The syntax is as follows:

```
WAIT FOR cond_expr
```

The ***cond_expr*** indicates a Condition expression. See Chap.8. - Condition Handlers in PDL2 Programming Language Manual, for further information about Conditions.

2.2.2 DELAY Statement

The DELAY statement causes execution of the program issuing it to be suspended for a specified period of time, expressed in milliseconds. Some events (such as current and pending motions, conditions scanning, etc.) continue even while a program is delayed.

```
DELAY int_expr
```

The ***int_expr*** indicates the time to delay in milliseconds.

2.2.3 HOLD Statement

The HOLD statement places all running holdable programs in a ready state and causes motion to decelerate to a stop.

```
HOLD
```

The HOLD statement works exactly like the HOLD button on the TP and control panel. START button must be pressed to place holdable programs back into a running state.

The HOLD statement can be used in both holdable and non-holdable programs.

2.2.4 CYCLE and EXIT CYCLE Statements

The CYCLE statement can either be a separate statement or an option on the BEGIN statement.

```
CYCLE
```

or

```
BEGIN CYCLE
```

It allows the programmer to create a continuous cycle. When the program END statement is encountered, execution continues back at the CYCLE statement.
The CYCLE statement is only allowed in the main program. The CYCLE statement cannot be used inside a routine. A program can contain only one CYCLE statement.

The EXIT CYCLE statement causes program execution to skip the remainder of the current cycle and immediately begin the next cycle. An exited cycle cannot be resumed. Exiting a CYCLE cancels all pending and current motions.
The EXIT CYCLE statement can be used in the main program as well as routines.
Exiting a cycle does NOT close files, detach resources, disable or purge condition handlers, or unlock arms. Consequently, it is more powerful than a simple GOTO Statement.

2.2.5 RETURN Statement

The RETURN statement is used to return program control from the routine currently being executed to the place where it was called.
For function routines, it also returns a value to the calling program or routine.

```
RETURN <(value)>
```

2.3 File Manipulation Statements

The basic PDL2 Statements that allow programs to manipulate files are:

– OPEN FILE Statement
– CLOSE FILE Statement
– READ Statement
– WRITE Statement.

LOGICAL UNIT NUMBERS - A logical unit number (LUN) represents a connection between a program and a physical device which the program can communicate with. The following predefined LUNs are recognized as already being opened for I/O operations:
– LUN_TP - Teach Pendant (TP:), which is the default LUN;
– LUN_CRT - WinC4G Terminal (CRT:)
– LUN_NULL - Null device (NULL:)

2.3.1 OPEN FILE Statement

The OPEN FILE statement opens a LUN on the specified device. This establishes a connection between the program and the device through which I/O operations can be performed.

```
OPEN FILE lun_var (device_str, access_str)
```

The ***lun_var*** can be either any INTEGER variable defined by the User or a predefined LUN.
The ***device_str*** can be any string expression representing an I/O device (a window, communication, or file device). File devices also include a file name and file extension. The default file device is ‘UD:’.
The ***access_str*** can be any string expression representing the access which the device is to be opened with:

```
R       -- read only
W       -- write only
RW      -- read and write
WA      -- write append
RWA     -- read and write append
```

Examples :

```
OPEN FILE file_lun (‘stats.dat’, ‘R’)  -- opens file stats.dat, read only
OPEN FILE comm_lun (‘COM1:’, ‘R’)  -- opens comm port COM1:, read only
```

2.3.2 CLOSE FILE Statement

The CLOSE FILE statement closes a LUN, ending the connection between the program and the device. Any buffered data is written to the device before the CLOSE FILE statement is executed.

```
CLOSE FILE lun_var
```

The ***lun_var*** can be either any INTEGER variable defined by the User or a predefined LUN, representing an open LUN. It is possible to specify the reserved word ***ALL*** instead of a LUN: this will close all files opened by the program.

2.3.3 READ Statement

The READ statement reads input data into program variables from the specified LUN.

The syntax of the READ statement is as follows:

```
READ <lun_var> (var_id <, var_id>...)
```

The ***lun_var*** can be either any INTEGER variable defined by the User or a predefined LUN. The default LUN is used if a ***lun_var*** is not specified.
The ***var_id***s are variable identifiers for the input data.

Example:

```
READ (body_type, total_units, operator_id) -- reads three values from the default LUN
```

2.3.4 WRITE Statement

The WRITE statement writes output data from a program to the specified LUN.

The syntax of the WRITE statement is as follows:

```
WRITE <lun_var> (expr <, expr>...)
```

The ***lun_var*** can be either any INTEGER variable defined by the User or a predefined LUN. The default LUN is used if a ***lun_var*** is not specified.
The ***expr*** can be any expression of INTEGER, REAL, BOOLEAN, STRING, POSITION, JOINTPOS data type.

Example:

```
WRITE ('Enter body style, units to process, and operator id.', NL)
```

The reserved word NL (new line) can also be used as a data item. NL causes a new line to be output. See the shown above example.

2.4 Motion Statements

This section describes the PDL2 statements that control arm motion: MOVE statements.

The syntax of the MOVE statement is as follows:

```
MOVE <arm_clause> <traj_clause> dest_clause <opt_clauses>
```

The following sections describe the shown above motion items:

– ARM Clause

– TRAJECTORY Clause

– DESTINATION Clause

– Optional Clauses.

2.4.1 ARM Clause

The ***arm_clause*** designates which arm is to be moved as part of a MOVE statement. For programs that control a single arm only, no designation is needed. If an arm clause is not included, the default arm is used. The programmer can designate a default arm as a program attribute in the PROGRAM statement, as follows:

```
PROGRAM armtest PROG_ARM=1
. . .
BEGIN
  MOVE TO dest             -- moves arm 1
  MOVE ARM[2] TO dest_2    -- moves to arm 2
END armtest
```

If a statement needs more than a single line, commas can be used to end a line after the destination clause or after each optional clause. The reserved word ENDMOVE must then be used to indicate the end of the statement.

2.4.2 TRAJECTORY Clause

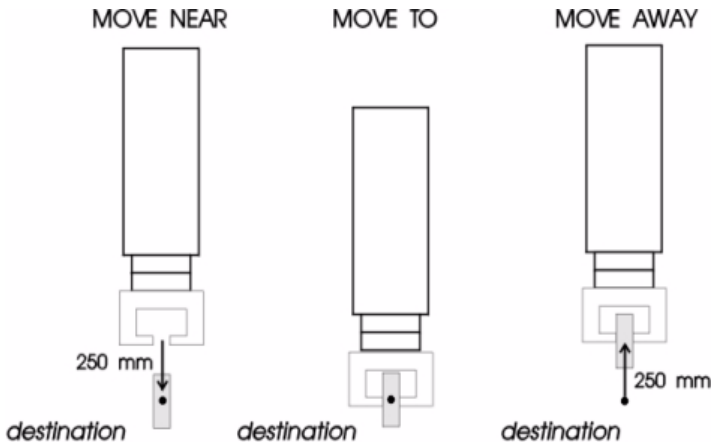
The ***traj_clause*** specifies the trajectory type: JOINT, LINEAR, CIRCULAR. The trajectory, when specified with the MOVE statement, only affects the motion for which it is designated.
If a trajectory clause is not included in the MOVE statement, the value of the predefined variable $MOVE_TYPE is used.
Examples:

```
$MOVE_TYPE := JOINT    -- assigns modal value
MOVE TO perch          -- joint move
MOVE LINEAR TO slot    -- linear move
MOVE TO perch          -- joint move
```

2.4.3 DESTINATION Clause

The ***dest_clause*** specifies the **kind** of MOVE and its **destination**. There are the following forms of the destination clause:

Fig. 1 - MOVE TO, NEAR and AWAY



- MOVE TO
- MOVE NEAR
- MOVE AWAY
- MOVE RELATIVE.

2.4.3.1 MOVE TO

MOVE TO moves the designated arm to a specified destination. See Fig. 1 - MOVE TO, NEAR and AWAY for further information. The destination can be any expression resulting in one of the following types:

– POSITION

– JOINTPOS.

Examples:

```
MOVE LINEAR TO pos(x, y, z, e1, e2, e3, config)
MOVE TO dest1
MOVE TO home
```

2.4.3.2 MOVE NEAR

MOVE NEAR allows the programmer to specify a destination along the tool approach vector that is within a specified distance from a position. The distance, specified as a real expression, is measured in millimeters along the negative tool approach vector. The destination can be any expression resulting in one of the following types:

– POSITION

– JOINTPOS.

See Fig. 1 - MOVE TO, NEAR and AWAY for further information.

Example:

```
MOVE NEAR destination BY 250.0
```

2.4.3.3 MOVE AWAY

MOVE AWAY allows the programmer to specify a destination along the tool approach vector that is a specified distance away from the current position. The distance, specified as a real expression, is measured in millimeters along the negative tool approach vector. See Fig. 1 - MOVE TO, NEAR and AWAY for further information.

Example:

```
MOVE AWAY 250.0
```

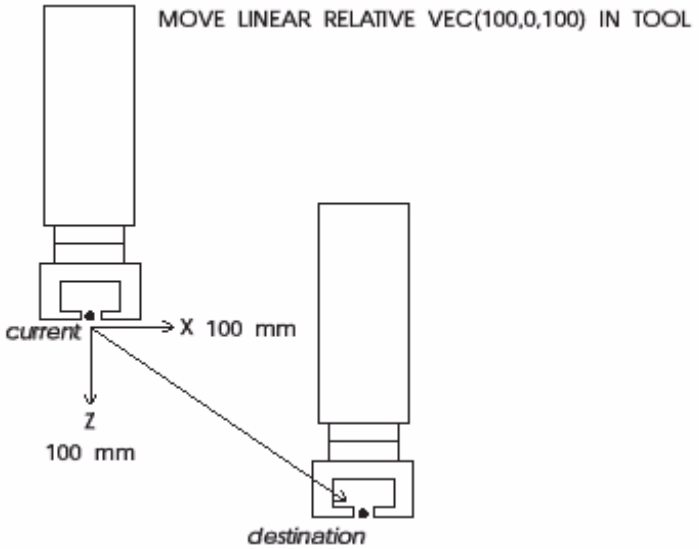
2.4.3.4 MOVE RELATIVE

MOVE RELATIVE allows the programmer to specify a destination relative to the current location of the arm. The destination is indicated by a vector expression, measured in millimeters, using the specified coordinate frame.

Example:

```
MOVE RELATIVE VEC(100, 0, 100) IN TOOL
```

The item following the reserved word IN is a frame specification which must be one of the predefined constants TOOL, BASE, or UFRAME.



2.4.4 Optional Clauses

Optional clauses can be used to provide more detailed instructions for the motion. An important optional clause is the WITH Clause, described in the next section.

2.4.4.1 WITH Clause

The optional WITH clause can specify temporary values for **predefined motion variables** which just apply for the duration of the motion.

The WITH clause only affects the motion caused by the current MOVE statement. Previous motions or those that follow are not affected.

The syntax of the WITH clause is as follows:

```
WITH designation <, designation>...
```

where ***designation*** means

```
predefined_motion_variable = value
```

For a full list of the allowed predefined motion variables, see par. 4.1.4.3 WITH Clause on page 4-8 in **PDL2 Programming Language** Manual.

Examples:

```
MOVE TO p1 WITH $PROG_SPD_OVR = 50 -- the program speed override is 50% just during the current motion

MOVE TO p1 WITH $ARM_DATA[1].ARM_SPD_OVR = 20, $TERM_TYPE = FINE -- the arm speed override is 20% and
                                                                  -- the type of motion termination is
                                                                  -- FINE, just during the current motion

MOVE TO p1 WITH $PROG_SPD_OVR = 50,  -- the program speed override is 50%,
    WITH $MOVE_TYPE = LINEAR,        -- the motion type is LINEAR,
    WITH $TOOL = drive_tool,         -- the reference frame is drive_tool, just during the current motion
ENDMOVE
```

2.5 Bit Manipulation Built-in Routines

The following are two predefined routines (referred to as built-in routines) made available by PDL2 language, to support bit manipulation.

– BIT_CLEAR Built-In Procedure

– BIT_SET Built-In Procedure

– BIT_TEST Built-In Function.

2.5.1 BIT_CLEAR Built-In Procedure

The BIT_CLEAR procedure clears (put to 0) ONE bit of either an INTEGER variable or a port.

```
BIT_CLEAR(var_ref, bit_num)
```

The user must indicate the following information:

– ***var_ref*** - the INTEGER variable or port name

– ***bit_num*** - an expression indicating the bit to be cleared (1..32)

Examples:

```
BIT_CLEAR(var1, 1) -- clears the 1st bit of variable var1
BIT_CLEAR(var2, bit_num) -- clears the bit specified in bit_num variable, of variable var2
BIT_CLEAR($WORD[4], 1) -- clears the 1st bit of the 4th $WORD predefined variable
```

2.5.2 BIT_SET Built-In Procedure

The BIT_SET procedure sets (put to 1) ONE bit of either an INTEGER variable or a port.

```
BIT_SET(var_ref, bit_num)
```

The user must indicate the following information:

– ***var_ref*** - the INTEGER variable or port name

– ***bit_num*** - an expression indicating the bit to be set (1..32)

Examples:

```
BIT_SET(var1, 1) -- sets the 1st bit of variable var1
BIT_SET(var2, bit_num) -- sets the bit specified in bit_num variable, of variable var2
BIT_SET($BYTE[2], 1) -- clears the 1st bit of the 2nd $BYTE predefined variable
```

2.5.3 BIT_TEST Built-In Function

The BIT_TEST Built-In function returns a BOOLEAN value indicating whether a bit of an INTEGER is set or cleared.

```
bool_var := BIT_TEST(test_val, bit_num <, bit_state>)
```

The user must indicate the following information:

– ***test_val*** - INTEGER value whose bit is to be tested. *test_val* can be an expression, a user defined variable reference, or a system port reference.

– ***bit_num*** - INTEGER value specifying the bit to be tested. The value must be in the range 1 to 32 where 1 corresponds to the least significant bit of the INTEGER.

– ***bit_state*** - BOOLEAN value indicating the desired bit state to test for. If not specified, ON is assumed.

– ***bool_var*** - BOOLEAN returned value: TRUE if the ***bit_num*** bit in ***test_val*** is currently set to ***bit_state***; FALSE if not.

Examples:

```
bool_var:= BIT_TEST($WORD[index], bit_num)
bool_var:= BIT_TEST(test_val, bit_num)
bool_var:= BIT_TEST(test_val, bit_num, FALSE)
```

2.6 ASSIGNMENT statement

The most common statement is the ASSIGNMENT statement. The ASSIGNMENT statement (:=) specifies a new value of a variable, using the result of an evaluated expression. The value resulting from the expression must be of the same data type as the variable.

The syntax is:

```
variable := expression
```

Basic Training Course - C4G USE AND PROGRAMMING - Appendix 2

– Sample Programs

1 - Example no.1

2 - Example no.2

3 - Example no.3

4 - Example no.4

5 - Example no.5

6 - Example no.6

7 - Example no.7

1 Example no.1

```
PROGRAM prog_0 EZ

--*** VARIABLES DECLARATION ***--

VAR scelta_prog : INTEGER -- VARIABILE FOR PROGRAM CHOICE

--*** Definition of ROUTINEs owned by current program ***--

ROUTINE call_prog_cambio_tool EXPORTED FROM prog_0
.....................................

ROUTINE call_prog_cambio_tool
BEGIN
.....................................

-- ----------------------- Main body ----------------------------

BEGIN
  call_prog_0
END prog_0
```

2 Example no.2

```
PROGRAM prog_esempio1 EZ, PROG_ARM = 1, STACK = 2048

--*** VARIABLES DECLARATION ***--

VAR jnt0001J, jnt0002J, jnt0003J, jnt0004J, jnt0005J : JOINTPOS FOR ARM[1]
  pnt0006P, pnt0007P, pnt0008P, pnt0009P, pnt0010P : POSITION
  pnt0011P, pnt0012P, pnt0013P, pnt0015P, pnt0014P : POSITION
  pnt0016P, pnt0017P, pnt0018P, pnt0019P, pnt0020P : POSITION

--*** Definition of ROUTINEs owned by current program ***--

ROUTINE call_prog_esempio1 EXPORTED FROM prog_esempio1
ROUTINE call_prog_esempio1
BEGIN
l_1::
  MOVE JOINT TO jnt0001J -- first move
  MOVE JOINT TO pnt0010P
  MOVE CIRCULAR TO pnt0013P VIA pnt0012P
  MOVE CIRCULAR TO pnt0015P VIA pnt0014P
  MOVEFLY LINEAR TO pnt0016p ADVANCE
  MOVEFLY CIRCULAR TO  pnt0017P VIA  pnt0018P ADVANCE
  MOVE JOINT TO pnt0019P
  MOVE JOINT TO jnt0002J -- last move
GOTO l_1
END call_prog_esempio1

-- ------------------------ Main body -----------------------------

BEGIN
  call_prog_esempio1
END prog_esempio1
```

3 Example no.3

```
PROGRAM prog_esempio2 EZ, PROG_ARM = 1, STACK = 2048

--*** VARIABLES DECLARATION ***--

VAR pnt0061P, pnt0062P, pnt0063P, pnt0064P, pnt0065P : POSITION
  pnt0056P, pnt0057P, pnt0058P, pnt0059P, pnt0060P : POSITION
  ....................................

--*** Definition of ROUTINEs owned by current program ***--

ROUTINE call_prog_esempio2 EXPORTED FROM prog_esempio2
ROUTINE call_prog_esempio2
BEGIN
l_1::
  MOVE JOINT TO pnt0064P -- prima move
  $ARM_ovr:=90 -- velocita impostata al 90%
  MOVE JOINT TO pnt0001P
  MOVEFLY JOINT TO pnt0050P ADVANCE
  MOVEFLY Linear TO pnt0052P ADVANCE
  $arm_ovr:= 30 -- velocita impostata al 30%
  MOVEFLY JOINT TO pnt0050P ADVANCE
  MOVE JOINT TO pnt0001P
  MOVEFLY linear TO pnt0050P ADVANCE
  MOVEFLY JOINT TO pnt0052P ADVANCE
  $arm_spd_ovr:= 60 -- velocita impostata al 90%
  MOVEFLY JOINT TO pnt0050P ADVANCE
  MOVE JOINT TO pnt0001P
  MOVEFLY Linear TO pnt0062P ADVANCE -- ultima move
 GOTO l_1
END call_prog_esempio2


-- ------------------------ Main body ----------------------------
BEGIN
  call_prog_esempio2
END prog_esempio2
```

4 Example no.4

```
PROGRAM prog_esempio3 EZ, PROG_ARM = 1, STACK = 2048

--*** VARIABLES DECLARATION ***--

VAR pnt0061P, pnt0062P, pnt0063P, pnt0064P, pnt0065P : POSITION
.....................................

--*** Definition of ROUTINEs owned by current program ***--

ROUTINE call_prog_esempio3 EXPORTED FROM prog_esempio3
ROUTINE call_prog_esempio3
BEGIN
l_8::
  MOVE JOINT TO pnt0064P
.....................................
GOTO l_8
END call_prog_esempio3

-- ------------------------ Main body ----------------------------
BEGIN
  call_prog_esempio3
END prog_esempio3
```

5 Example no.5

```
PROGRAM prog_esempio4 EZ, stack = 2048

--*** VARIABLES DECLARATION ***--

VAR pnt0001P : POSITION

--*** Definition of ROUTINEs imported from other programs ***--

ROUTINE call_prog_esempio1 EXPORTED FROM prog_esempio1
.....................................

--*** Definition of ROUTINEs owned by current program ***--

ROUTINE call_prog_esempio4
BEGIN
  MOVE JOINT TO pnt0001P
  WAIT FOR $FDIN[21] OR $FDIN[21] OR $FDIN[21]  -- wait for u1, u2, u3 being pressed
  IF $FDIN[21]=ON THEN -- with u1 call to prog_esempio1
    call_prog_esempio1
  ENDIF
  IF $fdin[22]=ON THEN -- with u2 call to prog_esempio2
    call_prog_esempio2
  ENDIF
  IF $fdin[23]=ON THEN -- with u3 call to prog_esempio3
    call_prog_esempio3
  ENDIF
END call_prog_esempio4

-- ------------------------ Main body -----------------------------
BEGIN
 CYCLE
 call_prog_esempio4
END prog_esempio4
```

6 Example no.6

```
PROGRAM comau PROG_ARM = 1, STACK = 2048

--*** VARIABLES DECLARATION ***--
VAR pnt0047P : POSITION
  pnt0042P, pnt0043P, pnt0044P, pnt0045P, pnt0046P : POSITION
  ..................................

--*** Definition of ROUTINEs imported from other programs ***--
  ROUTINE collision_end EXPORTED FROM collision
  ROUTINE collision_start(ai_coll, ai_tempo : INTEGER) EXPORTED FROM collision
  ..................................

--*** Definition of ROUTINEs owned by current program ***--
  ROUTINE apri_pinza    -- gun open ROUTINE --
  ..................................

-- ----------------------- Main body ----------------------------
BEGIN
  call_comau
END comau
```

6.1 Variables file for example no.6

```
-- File comau.lsv   16-Dec-08 11:29:27
-- SW Version  3.21
-- $PROP_AUTHOR:MU$
-- $PROP_DATE:09-Oct-08 15:06:02$
-- $PROP_VERSION:3.21.000$
-- $PROP_HOST:NS_759$
--
--$$   ARM: 1 AXES: 6 AUX:0 MASK: 63
--$$   ARM: 2 AXES: 7 AUX:0 MASK: 64

CONTATORE INT  Priv  1

DIST REA  Priv  40

jnt0001J JNTP Arm: 1 Ax: 6 Priv
 1:  47.16 2: -49.60 3:-103.00 4: -23.71 5:  95.96 6:  -5.24
     ..............................
```

7 Example no.7

PR_PLC Program manages:

– Opening/Closing clamps in automatic and manual
– Task reservation
– Request to enter access port
– Check of Robot home position + possible positioner
– Keyboard bright signals

EZ_ULIB Program manages:

– Connession/Disconnession PTDV external axes
– Dialogue of obstruction/out of obstruction robot-robot, robot-PTDV
– Arm1, Arm2 and PTDV variables and memories initialization
– Check for Arm1,Arm2 and PTDV home position

PROG_0 Program

– waits for an event and starts programs of job, torch cleaning, torch control, etc

Prog_1 Program and PROG_2 Program

– are two job sample programs

PROG_100 Program manages:

– Manual/Automatic PTDV rotation

PULIZIA_ARM_1 Program

– torch cleaning cycle

RIPOSO_ARM_1 Program

– moves the robot to the home position.

7.1 PR_PLC Program

```
PROGRAM pr_plc &PA, NOHOLD

--*** CONSTANTS DECLARATION ***--
CONST
  ki_puls_ripr = 9
  ki_barr_ripr = 10
  ......................

--*** RECORDs DEFINITION ***--
TYPE bloccaggio = RECORD
    descr : STRING[30]   -- 'bloccaggio' data type description
    ......................
ENDRECORD
  ......................

--*** VARIABLES DECLARATION ***--
VAR
  sequenza_ap : ARRAY[2, 10] OF passo_ap EXPORTED FROM pr_user
  sequenza_ch : ARRAY[2, 10] OF passo_ch EXPORTED FROM pr_user
  ......................

--*** Definition of ROUTINEs owned by current program ***--
ROUTINE check_pos_master : BOOLEAN
BEGIN
  ......................
END check_pos_master

ROUTINE check_posizionatore : INTEGER
  ......................

-------------------------- Main body -----------------------------
BEGIN
  ERR_TRAP_ON(39960)
  -- ru_def_condition   -- Condition definition
  ru_init   -- General Initialization

loop::
  ru_clock
  ......................
  DELAY 100
  GOTO loop
END pr_plc
```

7.2 EZ_ULIB Program

```
PROGRAM ez_ulib EZ, STACK = 2048

--*** VARIABLES DECLARATION ***--

VAR pnt0001j : JOINTPOS FOR ARM[1]
VAR
  ..........................

--*** Definition of ROUTINEs owned by current program ***--

ROUTINE in_area_arm1(num_area : INTEGER) EXPORTED FROM ez_ulib
ROUTINE in_area_arm2(num_area : INTEGER) EXPORTED FROM ez_ulib
..........................

ROUTINE ru_ins_dis_otturatore(ai_cmd : INTEGER)
VAR li_cmd : INTEGER
BEGIN
..........................
END ru_ins_dis_otturatore
ROUTINE ru_attiva_ax1_arm2   -- activation of SIK AX10
  ......................

------------------------- Main body ----------------------------

BEGIN
 ......................
END ez_ulib
```

7.3 PROG_0 Program

```
PROGRAM prog_0 EZ, PROG_ARM = 1, STACK = 2048

--*** VARIABLES DECLARATION ***--

VAR
  riposo_master : JOINTPOS FOR ARM[1] EXPORTED FROM ez_ulib

--*** Definition of ROUTINEs imported from other programs ***--

ROUTINE call_prog_4 EXPORTED FROM prog_4
  ..................................

--*** Definition of ROUTINEs owned by current program ***--

ROUTINE call_prog_0
-----------------------------------------------------
----------- robot Master handling Program --------
-----------------------------------------------------
BEGIN
  ..................................

END call_prog_0

------------------------- Main body ----------------------------

BEGIN
  call_prog_0
END prog_0
```

7.4 Prog_1 Program

```
PROGRAM prog_1 EZ, PROG_ARM = 1, STACK = 2048

--*** VARIABLES DECLARATION ***--

VAR xtn0092X, xtn0093X, xtn0094X, xtn0095X, xtn0096X : XTNDPOS FOR ARM[1]
  ..................................

--*** Definition of ROUTINEs owned by current program ***--

ROUTINE arc_star EXPORTED FROM prog_1
ROUTINE sub_1 EXPORTED FROM prog_1
.....................................
ROUTINE arc_star
.....................................

ROUTINE call_prog_1
.....................................
END call_prog_1

-------------------------- Main body -----------------------------

BEGIN
  call_prog_1
END prog_1
```

7.5 PROG_2 Program

```
PROGRAM prog_2 EZ, PROG_ARM = 1, STACK = 2048

--*** VARIABLES DECLARATION ***--

VAR xtn0096X, xtn0098X, xtn0099X : XTNDPOS FOR ARM[1]
  ..................................

--*** Definition of ROUTINEs owned by current program ***--

ROUTINE call_prog_2 EXPORTED FROM prog_2
  ..................................
ROUTINE call_prog_2
BEGIN
  ..................................
END call_prog_2

--------------------------- Main body -----------------------------

BEGIN
  call_prog_2
END prog_2
```

7.6 PROG_100 Program

```
PROGRAM prog_100 EZ, PROG_ARM = 2, STACK = 2048

--*** VARIABLES DECLARATION ***--

VAR
  lr_pos_target : REAL NOSAVE
VAR
  li_spd, li_acc, li_dec : INTEGER NOSAVE

--*** Definition of ROUTINEs owned by current program ***--

ROUTINE call_prog_100 EXPORTED FROM prog_100
  ......................................
ROUTINE ru_rotaz_auto
BEGIN
ROUTINE call_prog_100
BEGIN
.......................................
END call_prog_100

-------------------------- Main body ----------------------------

BEGIN
  call_prog_100
END prog_100
```

7.6.1 PULIZIA_ARM_1 Program

```
PROGRAM pulizia_arm1 EZ, PROG_ARM = 1, STACK = 2048

--*** VARIABLES DECLARATION ***--

VAR pnt0007p, pnt0008p, pnt0009p, pnt0001p, pnt0010p : POSITION
.............................

--*** Definition of ROUTINEs imported from other programs ***--

ROUTINE al_wire_frw(ai_time_wire_frw, ai_speed_wire_frw, ai_status : INTEGER) EXPORTED FROM al_user
.............................

--*** Definition of ROUTINEs owned by current program ***--

ROUTINE call_pulizia_arm1
BEGIN
  ..........................

-------------------------- Main body ----------------------------

BEGIN
  call_pulizia_arm1
END pulizia_arm1
```

7.6.2 RIPOSO_ARM_1 Program

```
PROGRAM riposo_arm1 EZ, STACK = 2048

--*** VARIABLES DECLARATION ***--

VAR xtn0001x, xtn0002x : XTNDPOS FOR ARM[1]

--*** Definition of ROUTINEs owned by current program ***--

ROUTINE ru_allarmi(li_allarmi : INTEGER) EXPORTED FROM ez_ulib   -- Errors display ROUTINE
.....................................
ROUTINE call_riposo_arm1
BEGIN
  ---------------------------------------------
  ----------- Sleeping down Program ----------
  -- Robot Master / Slave / Positioner -------
  ---------------------------------------------

..................

-------------------------- Main body -----------------------------

BEGIN
  call_riposo_arm1
END riposo_arm1
```

Basic Training Course - C4G USE AND PROGRAMMING - Appendix 3

1 Exercizes

1. Robot generic movement description.
2. Robot movement in Joint, Tool, Base and Uframe.
3. Tool automatic calculation (with the two methods)
4. UFrame automatic calculation.
5. Creation of Motion programs with different types of movement:
   - Linear
   - Joint
   - Circular
   - Create a position with a JNT variable (home position)
6. Creation of a program with different FLY positions and different speeds to understand the Robot behaviour in a FLY position at different speeds.
7. Creation of a NO HOLDABLE program using the U1 U2 U3 U4 keys and the corresponding leds
8. INPUT/OUTPUT setting
9. Creation of a manipulation program using the previously set INPUT and OUTPUT.
10. Creation of a No HOLDABLE program that waits for a consent from the operator (for example using U1 key) and, once received the consent, it calls the previously created manipulation program.
11. Modification of the manipulation program, on WinC4g, inserting $ARM_SPD_OVR, $TERM_TYPE, $FLY_PER etc.
12. Moving the plan of job and calculating again the Uframe of the manipulation program.
13. Where it is allowed, do the same test with the Tool (either modifying or moving it)
14. Creation of a program on WINC4G, load it on the robot and assign the physical position to the program points.
15. Configuration of either Profibus or DeviceNet board (both from TP and WinC4G)
16. Creation of an Offline program where ROUTINES, CONSTANTS and VARIABLES are used (it is possible to modify the previously created manipulation program) - see. par. 1.1 Data items on page 6-2 and par. 1.3 Routines on page 6-5 nel Chap. Basic Training Course - C4G USE AND PROGRAMMING - Appendix 1:
   – in the program, try to use the PDL2 language basic statements (IF, WAIT FOR, CYCLE, GO TO, etc) - see par. 2 Basic PDL2 Statements on page 6-6 in Chap. Basic Training Course - C4G USE AND PROGRAMMING - Appendix 1.

Comau Robotics After Sales Service

Monday-Friday 8.00 - 18.00
(After hours service by contract)

Tel. +39-011-0045553
Fax +39-011-0045416
service.robotics@comau.com

After Sales

Director
Mussotto Stefano
Tel. +39-011-0045.489
stefano.mussotto@comau.com

Tecnical Service - Italy

Manager
Lombardi Ruggero
Tel. +39-011-0045.479
ruggero.lombardi@comau.com

P.O. Management
Papini Cristina
Tel. +39-011-0045.553
cristina.papini@comau.com

Italian South Site
Paolocci Ciro
Tel. +39-081-8034073
Fax: +39-081-8034072
ciro.paolocci@comau.com

Partner Support (Italy)
Bullio Claudio
Tel. +39-011-0045.570
claudio.bullio@comau.com

Maintenance Contract and Reused Solutions
Carda Davide
Tel. +39-011-0045.528
davide.carda@comau.com

Technical Service - Rest of World

Manager
Gastaldi Gianluca
Tel. +39-011-0045.633
gianluca.gastaldi@comau.com

Partner support (Rest of World)
Taramino Marco
Tel. +39-011-0049.168
marco.taramino@comau.com

Spare and Reparation

Manager
Taberna Paolo
Tel. +39-011-0045.451
paolo.taberna@comau.com

P.O. Management
Mineo Alberto
Tel. +39-011-0045.478
spares.robotics@comau.com

Training

Manager
Menon Massimo
Tel. +39-011-0045.453
massimo.menon@comau.com