Comau Robotics
Instruction Handbook

C4G OPEN

System Software Rel. 3.2x

CR00757550_en-05/1208

COMAU ROBOTICS

SUMMARY

PREFACE

Symbols used in the manual

The symbols for **WARNING, CAUTION** and **NOTES** are indicated below together with their significance.

This symbol indicates operating procedures, technical information and precautions that if ignored and/or are not performed correctly could cause injuries.

This symbol indicates operating procedures, technical information and precautions that if ignored and/or are not performed correctly could cause damage to the equipment.

This symbol indicates operating procedures, technical information and precautions that it are important to highlight.

Reference documents

This document refers to the **C4G Control Unit**.

The complete set of manuals for the **C4G** consists of:

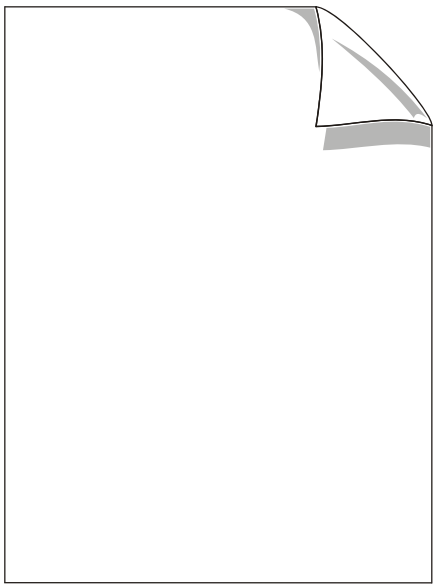
Comau	C4G Control Unit	– Technical Specifications – Transport and installation – Guide to integration, safeties, I/O and communications – C4G Control Unit Use.

These manuals are to be integrated with the following documents:

Comau	Robot	– Technical Specifications – Transport and installation – Maintenance
	Programming	– PDL2 Programming Language Manual – VP2 - Visual PDL2 – Motion programming
	Applications	– According to the required type of application.
Sintesi	Programmazione	– C4G Open Library - User's Guide

Modification History

– In 03/0908 version revision, the following modifications have been made:
  - new chapter Safety Precautions for C4G Open - pointed out carefully the need for the user to operate outside the Protected Area, at a safe distance from the robot;
  - Mode 0' - reserved to COMAU;
  - Kernel installation procedure - description modified
  - Mode 5 and Mode 7 - description upgrade.



1. GENERAL SAFETY PRECAUTIONS

1.1 Responsibilities

– The system integrator is responsible for ensuring that the Robot and Control System are installed and handled in accordance with the Safety Standards in force in the country where the installation takes place. The application and use of the protection and safety devices necessary, the issuing of declarations of conformity and any CE markings of the system are the responsibility of the Integrator.

– 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 originals or that have not been defined as equivalent by COMAU Robotics & Service

– The application of these Safety Precautions is the responsibility of the persons assigned to direct / supervise the activities indicated in the Applicability section,They are to make sure that the Authorised Personnel is aware of and scrupulously follow the precautions contained in this document as well as the Safety Standards in addition to the Safety Standards in force in the country in which it is installed.

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

The installation shall be made by qualified installation Personnel and should conform to all national and local codes.

1.2 Safety Precautions

1.2.1 Purpose

These safety precautions are aimed to define the behaviour and rules to be observed when performing the activities listed in the Applicability section.

1.2.2 Definitions

Robot and Control System
The Robot and Control System consists of all the functions that cover: Control Unit, robot, hand held programming unit and any options.

Protected Area
The protected area is the zone confined by the safety barriers and to be used for the installation and operation of the robot

Authorised Personnel
Authorised personnel defines the group of persons who have been trained and assigned to carry out the activities listed in the Applicability section.

Assigned Personnel
The persons assigned to direct or supervise the activities of the workers referred to in the paragraph above.

Installation and Putting into Service
The installation is intended as the mechanical, electrical and software integration of the Robot and Control System in any environment that requires controlled movement of robot axes, in compliance with the safety requirements of the country where the system is installed.

Programming Mode
Operating mode under the control of the operator, that excludes automatic operation and allows the following activities: manual handling of robot axes and programming of work cycles at low speed, programmed cycle testing at low speed and, when allowed, at the working speed.

Auto / Remote Automatic Mode
Operating mode in which the robot autonomously executes the programmed cycle at the work speed, with the operators outside the protected area, with the safety barriers closed and the safety circuit activated, with local (located outside the protected area) or remote start/stop.

Maintenance and Repairs
Maintenance and repairs are activities that involve periodical checking and / or replacement (mechanical, electrical, software) of Robot and Control System parts or components, and trouble shooting, that terminates when the Robot and Control System has been reset to its original project functional condition.

Putting Out of Service and Dismantling
Putting out of service defines the activities involved in the mechanical and electrical removal of the Robot and Control System from a production unit or from an environment in which it was under study.
Dismantling consists of the demolition and dismantling of the components that make up the Robot and Control System.

Integrator
The integrator is the professional expert responsible for the installation and putting into service of the Robot and Control System.

Incorrect Use
Incorrect use is when the system is used in a manner other than that specified in the Technical Documentation.

Range of Action
The robot range of action is the enveloping volume of the area occupied by the robot and its fixtures during movement in space.

1.2.3 Applicability

These Specifications are to be applied when executing the following activities:

– Installation and Putting into Service;
– Programming Mode;
– Auto / Remote Automatic Mode;
– Robot axes release;
– Stop distances (threshold values)
– Maintenance and Repairs;
– Putting Out of Service and Dismantling

1.2.4 Operating Modes

Installation and Putting into Service

- Putting into service is only possible when the Robot and Control System has been correctly and completely installed.
- The system installation and putting into service is exclusively the task of the authorised personnel.
- The system installation and putting into service is only permitted inside a protected area of an adequate size to house the robot and the fixtures it is outfitted with, without passing beyond the safety barriers. It is also necessary to check that under normal robot movement conditions there is no collision with parts inside the protected area (structural columns, power supply lines, etc.) or with the barriers. If necessary, limit the robot working areas with mechanical hard stop (see optional assemblies).
- Any fixed robot control protections are to be located outside the protected area and in a point where there is a full view of the robot movements.
- The robot installation area is to be as free as possible from materials that could impede or limit visibility.
- During installation the robot and the Control Unit are to be handled as described in the product Technical Documentation; if lifting is necessary, check that the eye-bolts are fixed securely and use only adequate slings and equipment.
- Secure the robot to the support, with all the bolts and pins foreseen, tightened to the torque indicated in the product Technical Documentation.
- If present, remove the fastening brackets from the axes and check that the fixing of the robot fixture is secured correctly.
- Check that the robot guards are correctly secured and that there are no moving or loose parts. Check that the Control Unit components are intact.
- If applicable, connect the robot pneumatic system to the air distribution line paying attention to set the system to the specified pressure value: a wrong setting of the pressure system influences correct robot movement.
- Install filters on the pneumatic system to collect any condensation.
- Install the Control Unit outside the protected area: the Control Unit is not to be used to form part of the fencing.
- Check that the voltage value of the mains is consistent with that indicated on the plate of the Control Unit.
- Before electrically connecting the Control Unit, check that the circuit breaker on the mains is locked in open position.
- Connection between the Control Unit and the three-phase supply mains at the works, is to be with a four-pole (3 phases + earth) armoured cable dimensioned appropriately for the power installed on the Control Unit. See the product Technical Documentation.
- The power supply cable is to enter the Control Unit through the specific fairlead and be properly clamped.
- Connect the earth conductor (PE) then connect the power conductors to the main switch.

- Connect the power supply cable, first connecting the earth conductor to the circuit breaker on the mains line, after checking with a tester that the circuit breaker terminals are not powered. Connect the cable armouring to the earth.
- Connect the signals and power cables between the Control Unit and the robot.
- Connect the robot to earth or to the Control Unit or to a nearby earth socket.
- Check that the Control Unit door (or doors) is/are locked with the key.
- A wrong connection of the connectors could cause permanent damage to the Control Unit components.
- The C4G Control Unit manages internally the main safety interlocks (gates, enabling pushbuttons, etc.). Connect the C4G Control Unit safety interlocks to the line safety circuits, taking care to connect them as required by the Safety standards. The safety of the interlock signals coming from the transfer line (emrgency stop, gates safey devices etc) i.e. the realisation of correct and safe circuits, is the responsibility of the Robot and Control System integrator.

In the cell/line emergency stop circuit the contacts must be included of the control unit emergency stop buttons, which are on X30. The push buttons are not interlocked in the emergency stop circuit of the Control Unit.

- The safety of the system cannot be guaranteed if these interlocks are wrongly executed, incomplete or missing.
- The safety circuit executes a controlled stop (IEC 60204-1 , class 1 stop) for the safety inputs Auto Stop/ General Stop and Emergency Stop. The controlled stop is only active in Automatic states; in Programming the power is cut out (power contactors open) immediately. The procedure for the selection of the controlled stop time (that can be set on ESK board) is contained in the Installation manual .
- When preparing protection barriers, especially light barriers and access doors, bear in mind that the robot stop times and distances are according to the stop category (0 or 1) and the weight of the robot..

Check that the controlled stop time is consistent with the type of Robot connected to the Control Unit. The stop time is selected using selector switches SW1 and SW2 on the ESK board.

- Check that the environment and working conditions are within the range specified in the specific product Technical Documentation.
- The calibration operations are to be carried out with great care, as indicated in the Technical Documentation of the specific product, and are to be concluded checking the correct position of the machine.
- To load or update the system software (for example after replacing boards), use only the original software handed over by COMAU Robotics & Service. Scrupulously follow the system software uploading procedure described in the Technical Documentation supplied with the specific product. After uploading, always make some tests moving the robot at slow speed and remaining outside the protected area.
- Check that the barriers of the protected area are correctly positioned.

Programming Mode

- 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.
- When possible the programming should be controlled from outside the protected area.
- Before operating inside the Protected Area, the operator must make sure from outside that all the necessary protections and safety devices are present and in working order, and especially that the hand-held programming unit functions correctly (slow speed, emergency stop, enabling device, etc.).
- During the programming session, only the operator with the hand-held terminal is allowed inside the Protected Area.
- If the presence of a second operator in the working area is necessary when checking the program, this person must have an enabling device interlocked with the safety devices.
- 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.
- When programming, the operator is to keep at a distance from the robot to be able to avoid any irregular machine movements, and in any case in a position to avoid the risk of being trapped between the robot and structural parts (columns, barriers, etc.), or between movable parts of the actual robot.
- When programming, the operator is to avoid remaining in a position where parts of the robot, pulled by gravity, could execute downward movements, or move upwards or sideways (when installed on a sloped plane).
- Testing a programmed cycle at working speed with the operator inside the protected area, in some situations where a close visual check is necessary, is only to be carried out after a complete test cycle at slow speed has been executed. The test is to be controlled from a safe distance.
- 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.
- During the first running of a new program, the robot may move along a path that is not the one expected.
- The modification of program steps (such as moving by a step from one point to another of the flow, wrong recording of a step, modification of the robot position out of the path that links two steps of the program), could give rise to movements not envisaged by the operator when testing the program.
- In both cases operate cautiously, always remaining out of the robot's range of action and test the cycle at slow speed.

Auto / Remote Automatic Mode

– 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.

– The operator can only activate automatic operation after having checked:
  • that the Robot and Control System is not in maintenance or being repaired;
  • the safety barriers are correctly positioned;
  • that there is nobody inside the protected area;
  • that the Control Unit doors are closed and locked;
  • that the safety devices (emergency stop, safety barrier devices) are functioning;

– Special attention is to be paid when selecting the automatic-remote mode, where the line PLC can perform automatic operations to switch on motors and start the program.

Robot axes release

– In the absence of motive power, the robot axes movement is possible by means of optional release devices and suitable lifting devices. Such devices only enable the brake deactivation of each axis. In this case, all the system safety devices (including the emergency stop and the enable button) are cut out; also the robot axes can move upwards or downwards because of the force generated by the balancing system, or the force of gravity.

Before using the manual release devices, it is strongly recommended to sling the robot, or hook to an overhead travelling crane.

Stop distances (threshold values)

– As for the stop distance threshold values for each robot type, please turn to the COMAU Robotics & Service Dept.

– Example: Considering the robot in automatic mode, in conditions of maximum extension, maximum load and maximum speed, when the stop pushbutton is pressed (red mushroom head pushbutton on WiTP) an NJ 370-2.7 Robot will stop completely in approx. 85° of motion, equivalent to approx. 3000 mm displacement measured on the TCP flange. Under these conditions indicated, the stoppage time of the NJ 370-2.7 Robot is 1.5 seconds.

– Considering the robot in programming mode (T1), when the stop pushbutton is pressed (red mushroom head pushbutton on WiTP) an NJ 370-2.7 Robot will stop completely in approx. 0.5 seconds.

Maintenance and Repairs

– When assembled in COMAU Robotics & Service, the robot is supplied with lubricant that does not contain substances harmful to health, however, in some cases, repeated and prolonged exposure to the product could cause skin irritation, or if swallowed, indisposition.
First Aid. Contact with the eyes or the skin: wash the contaminated zones with abundant water; if the irritation persists, consult a doctor.

If swallowed, do not provoke vomiting or take anything by mouth, see a doctor as soon as possible.

– Maintenance, trouble-shooting and repairs are only to be carried out by authorised personnel.

– When carrying out maintenance and repairs, the specific warning sign is to be placed on the control panel of the Control Unit, stating that maintenance is in progress and it is only to be removed after the operation has been completely finished - even if it should be temporarily suspended.

– Maintenance operations and replacement of components or the Control Unit are to be carried out with the main switch in open position and locked with a padlock.

– Even if the Control Unit is not powered (main switch open), there may be interconnected voltages coming from connections to peripheral units or external power sources (e.g. 24 Vdc inputs/outputs). Cut out external sources when operating on parts of the system that are involved.

– Removal of panels, protection shields, grids, etc. is only allowed with the main switch open and padlocked.

– Faulty components are to be replaced with others having the same code, or equivalent components defined by COMAU Robotics & Service.

After replacement of the ESK module, check on the new module that the setting of the stop time on selector switches SW1 and SW2 is consistent with the type of Robot connected to the Control Unit.

– Trouble-shooting and maintenance activities are to be executed, when possible, outside the protected area.

– Trouble-shooting executed on the control is to be carried out, when possible without power supply.

– Should it be necessary, during trouble-shooting, to intervene with the Control Unit powered, all the precautions specified by Safety Standards are to be observed when operating with hazardous voltages present.

– Trouble-shooting on the robot is to be carried out with the power supply cut out (Drive off).

– At the end of the maintenance and trouble-shooting operations, all deactivated safety devices are to be reset (panels, protection shields, interlocks, etc.).

– Maintenance, repairs and trouble-shooting operations are to be concluded checking the correct operation of the Robot and Control System and all the safety devices, executed from outside the protected area.

– When loading the software (for example after replacing electronic boards) the original software handed over by COMAU Robotics & Service is to be used. Scrupulously follow the system software loading procedure described in the specific product Technical Documentation; after loading always run a test cycle to make sure, remaining outside the protected area

– Disassembly of robot components (motors, balancing cylinders, etc.) may cause uncontrolled movements of the axes in any direction: before starting a disassembly procedure, consult the warning plates applied to the robot and the Technical Documentation supplied.

– It is strictly forbidden to remove the protective covering of the robot springs.

Putting Out of Service and Dismantling

– Putting out of service and dismantling the Robot and Control System is only to be carried out by Authorised Personnel.

– Bring the robot to transport position and fit the axis clamping brackets (where applicable) consulting the plate applied on the robot and the robot Technical Documentation.

– Before stating to put out of service, the mains voltage to the Control Unit must be cut out (switch off the circuit breaker on the mains distribution line and lock it in open position).

– After using the specific instrument to check there is no voltage on the terminals, disconnect the power supply cable from the circuit breaker on the distribution line, first disconnecting the power conductors, then the earth. Disconnect the power supply cable from the Control Unit and remove it.

– First disconnect the connection cables between the robot and the Control Unit, then the earth cable.

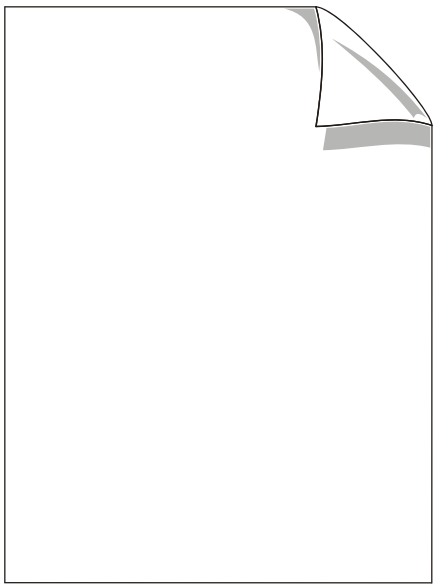
– If present, disconnect the robot pneumatic system from the air distribution line.

– Check that the robot is properly balanced and if necessary sling it correctly, then remove the robot securing bolts from the support.

– Remove the robot and the Control Unit from the work area, applying the rules indicated in the products Technical Documentation; if lifting is necessary, check the correct fastening of the eye-bolts and use appropriate slings and equipment only.

– Before starting dismantling operations (disassembly, demolition and disposal) of the Robot and Control System components, contact COMAU Robotics & Service, or one of its branches, who will indicate, according to the type of robot and Control Unit, the operating methods in accordance with safety principles and safeguarding the environment.

– The waste disposal operations are to be carried out complying with the legislation of the country where the Robot and Control System is installed.

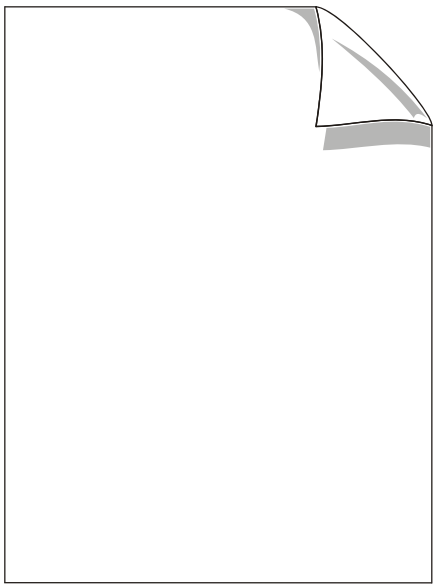


2. SAFETY PRECAUTIONS FOR C4G OPEN

To properly use the C4G Open System, please refer to the Safety Precautions described in Chap.1. - General Safety Precautions.

Particularly, we'd like to outline that since the C4G Open Control System is a powerful rapid research/prototyping system, it is mandatory to take the greatest care in moving the robot, because any robot moving directly comes from the external PC.
Furthermore, be sure that during cycle testing no operators are in the robot working area and any safety precautions have been performed, according to the legal restrictions in force in the Country where the System is used.
COMAU shall in no way be held liable for any accidents caused by incorrect or improper use of the System.
Thus, it is pointed out that the operator must always stay outside the Protected Area and the work cell must be provided with any safety devices, according to the legal restrictions (e.g. interlock safety gates).

It is needed to remark that the C4G Open System is properly working only when the Modal Selector Switch is in the AUTO position.



3. INTRODUCTION

Nowadays, universities and factories, more frequently use development and “rapid prototyping” systems. To do that, emerging tchnologies are used which allow to examine, to analitically and quickly evaluate new ideas, to develop innovative products and to decrease their time-to-market.

In the car industry, the use of “rapid prototyping” technologies is by now a well-known practice. Time-to-market is one of the success critical factors, in a very competitive market.

The three most important features related to the chosen devices and technologies to obtain such a target, are:

– reliability and safety,

– easiness to use,

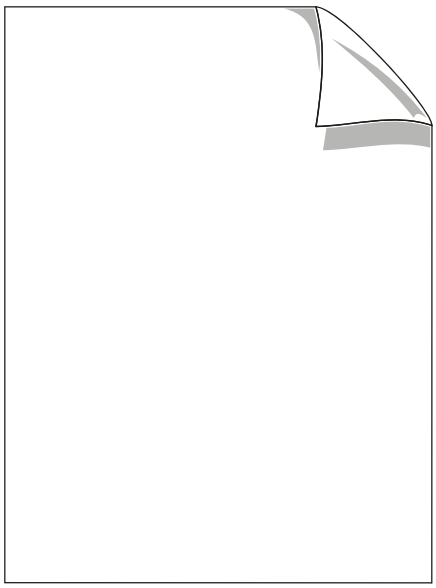
– flexibility in defining and creating new strategies, to control the manufacturing process.

The aim of the current document is to describe the “***Open***" evolution of the COMAU C4G Controller for industrial robots: i.e. the software option called "***C4G OPEN***".

C4G Open is a very innovative software and hardware control architecture which allows easy and safe integration of the Robot Control Unit whith an external Personal Computer, in order to help programming the automatized robotic cell at several levels, with the possibility of getting information from external sensors.

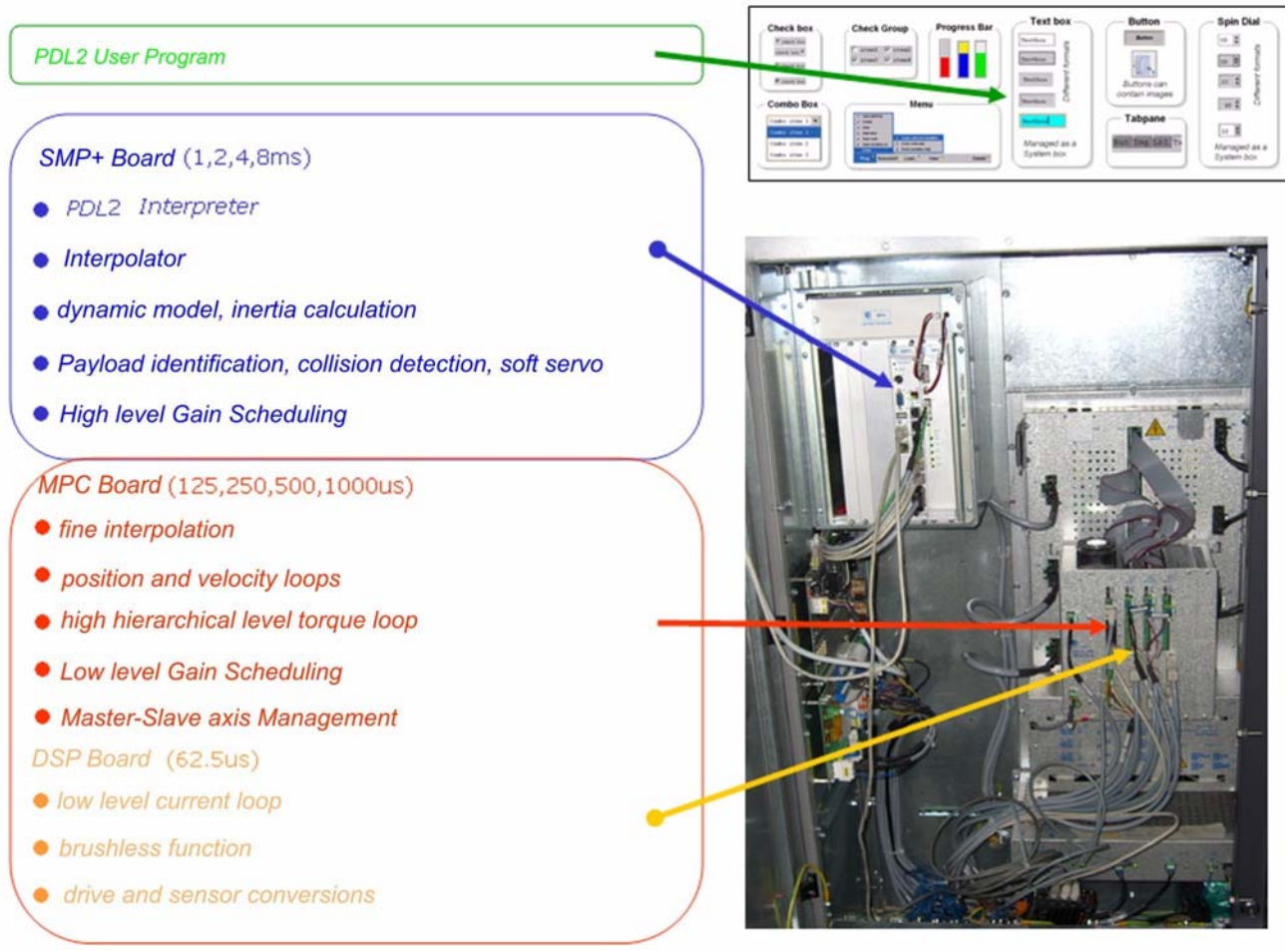
This control architecture has been conceived to quickly and low costs make up new robotic cells, provided with innovative robot control functionalities and advanced robotic applications.

Using a net of external sensors, without any restrictions on the type and mode of use, overcomes typical problems such as the “closure” of industrial robots control systems and offers to the users a wide range of access modes. For this reason C4G Open is considered as a reliable and safe working environment, as well as very interesting both for the academic world and for industries (robotic cells integrators, innovative robotic companies, etc.).



4. C4G SYSTEM

Fig. 4.1 - C4G System



As shown in Fig. 4.1, the Comau C4G manufacturing System includes the following software and hardware components:.

– PDL2 interface for robot programming

– SMP+ control board, on which all high level processes are implemented :
  - user programs interpretation;
  - operator interface handling (remote too);
  - network communications handling;
  - trajectory generation;
  - dynamic model calculation for the manipulator and tasks related to it;
  - collision detection and system diagnostics;
  - centralized high hierarchical level control (adaptive control handling);
  - axes synchronizing control;
  - handling all I/O devices;

– MPC control board, inside the multiaxis drive, which implements:
  - the fine interpolation of the robot trajectory;

- adaptive control of the robot position which includes 3 levels of control, interacting among themselves and the centralized control level;
- robot position control adaptation logics, according to the information available at centralized level;
- real time level of the system diagnostics;
- master-slave axes management;

– DSP control boards, within the multiaxis drive, which implements:
- control of the motor currents and torque generation process (independent axes control);
- power stage management;
- position sensor management and measures acquisition about angular position of the motor;
- high speed digital and analog I/Os handling.

The user can use the C4G Controller, by writing in a file the statements which describe the job to be performed by the robot. The available instructions are the ones provided by a specific Pascal-like programming language which is called PDL2.

The PDL2 programs are interpreted by the System Software, available on the SMP+ board (System Master Processor Plus). If a statement requires a motion, the SMP+ sofware interprets the general features of that motion, such as the starting and ending positions and the required speed. Then the trajectory is generated in order to perform the required movement.

Inside the MPC board (Motorola Power PC) the position and velocity loops are executed, which must calculate the reference current that allows the robot to perform the trajectory calculated by the trajectory generator.

The risulting information from MPC board are taken as references to be used by DSP boards which supply currents to the motors, in order to finally move the robot.

The DSA (Digital Servo Amplifier) device is composed by MPC and DSP boards. In the current document, any references to DSA will be made without specifying whether talking about MPC or DSP.

5. C4G OPEN SYSTEM

C4G Open is an improvement of the industrial robt controller; it is an optional feature of the standard product, based upon a real time communication in which an external Personal Computer interacts with the robot Controller at different levels, in the robot control (servo system) and in the trajectory generation.

The external contribution cooperates with, integrates or substitutes the industrial control system; however the standard controller always has an active role, during the process. This allows a reliable and safe management in both the system diagnotics and the execution of one or more application programs.

By means of the innovation of the "***open controller***", the external hardware structure and the industrial Control Unit, set up a new powerful control architecture for developing advanced robotic applications.

The applications which can be implemented by means of such a System, should be considered as not completely conventional advanced robot controller operations. Example:

– a sensors network, handled by the PC which executes a control action or a high level trajectory generation too,

– a cooperating video cameras and force sensors network, to complete mathematics and algorythms which control the robot arm motion.

The C4G Open architecture is based on a real time communication on Ethernet network which uses an owner UDP protocol, between **SMP+** board (network "client") and **MPC** board (acting as a "server").
The opening of the robot controller has been achieved by inserting a second "server" process which is the **PC** connected on the network (see Fig. 5.1):

– "SMP+" block includes the C4G Basic Operating System, the PDL2 interpreter and the trajectory generator,

– "DSA" includes the software which controls the position, velocity and current loops,

– "PC" is the external device which interfaces to the COMAU System,

– Switch is a hardware switch allowing the communication with "SMP+", "DSA" and "PC" blocks.

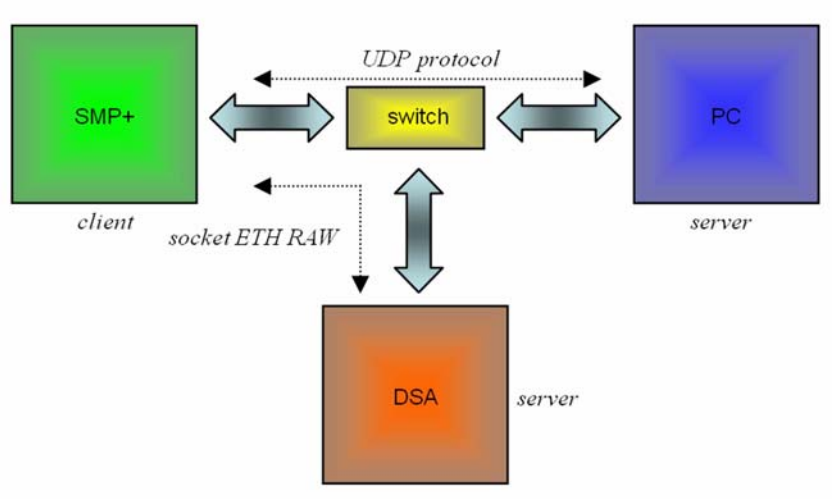
In current documentation, the word SMP+ stands for the software included in the SMP+ board, which is supposed to carry out the communication with DSA and PC.
The remaining part of the SMP+ software, is referred to as the "trajectory generator".

The "SMP+"-"DSA" communication protocol is a dedicated non-standard protocol, to optimize the data flow; it is called RAW.

The "SMP+"-"PC" communication protocol is a classical UDP protocol. The choice of using an UDP protocol for C4G to communicate with external devices, is because such a protocol is general purpose and moreover, COMAU already had operating

experiences of interfacing the C4G controller to drives which were different from DSA.

Fig. 5.1 - Open Controller



Not all the configurations based upon the industrial C4G System, can be created in the C4G Open System.
In the shown below table, the software and hardware C4G Open restrictions are listed:

Tab. 5.1 - C4G Open

Max number of ARMs	**4**
Max number of DSAs	**2**
Max number of axes	**20**
Max number of axes (for each ARM, including auxiliary axes)	**10**
Max number of auxiliary axes (for each ARM)	**4**

6. SYSTEM INITIAL CONFIGURATION

To be able to use C4G in Open Modality, first of all it is needed to properly configure the system.

The configuration is performed, following the listed below steps:

a. execute a PDL2 instructions set, to setup the Open Modality.
   The PDL2 setup instructions are as follows::
   - **$ARM_DATA[i].C4GOPEN_JNT_MASK** : a scalar including the mask of the i-th ARM axes, which are to be used with the Open Modality,
   - **$ARM_DATA[i].C4GOPEN_MODE[j]** : a vector including, for each axis, the being used Open Modality.

b. save the system configuration. Depending on issuing the command either from PDL2 program or from TP, act as follows:
   – from PDL2 program - command **CSA**: CONFIGURE SAVE ALL
   – from TP - open the **SETUP Page**, press **Config.** (**F1**), select item **1** - **Save in the configuration file (Save)**.

c. System restart - Depending on issuing the command either from PDL2 program or from TP, act as follows:
   – from PDL2 program - command **CCRC**: CONFIGURE CONTROLLER RESTART COLD
   – from TP - open the **Home Page** press **Restart** (**F4**) and select item **1** - **Cold**.

When restarting, if the specific robot is allowed to be controlled in open modality, and if on the related controller the C4G Open software option has been activated, the System is prepared to communicate with the external device.

Note that the stand-by functionality is not supported.

Anyway, the selected modality doesn't lock the real time communication which has been setup at the system restart: the user is allowed to dynamically modify it between a movement and the next one, also temporarily setting the modality which gives back to C4G the total control of the system.
The only feature which is definitively set up, is the mask of the axes to be controlled in open modality. It is set during the configuration phase and it cannot be modified until the next system restart.

After the system configuration for the open modality, when the system restarts, a check is performed on the mask. If some of the specified axes in the mask, are not defined in the characterization, the software simply ignores them.

If an axis is not included in an open controller, it is not considered when creating the communication buffer: this means that the communnication buffer only includes the axes which are configured as OPEN; it is up to the sender and the receiver to guarantee the proper mapping between the axis position in the buffer and the being controlled axis index.

This allows to optimize the communication between SMP+ and PC, because each data packet length doesn't include empty fields related to axes which are not interested in the

open modality. Obviously, if at first the user doesn't know which axes are to be controlled in OPEN modality, it is possible to declare all the axes in open modality and then to change the control strategy by means of a suitable field of the communication packet (see.Chap.8. - Communication Packet).

As far as mapping the position of the axis in the communication buffer, and the being controlled axis index, it is useful to introduce the following glossary, associated to the axes indexes, depending on the context:

– the **Logical** axes indexes are the ones directely seen by the operator, the ones the C4G user interface refers to;

– the **Physical** axes indexes are used to motors wiring, so they are referred to the configuration of the drive, the motors and the system wiring;

– the **RAW** axes indexes are used while building the communication packets between SMP+ and DSA (the packet SMP+ sends to DSA is called BLIP, whilst the one sent from DSA to SMP+ is called RAAZ). It is allowed to define up to two DSA, so some axes are referred to DSA 1 and the others to DSA 2. To sort the values related to the axes belonging to the two DSAs and to build two different packets, the RAW axes indexes must be obtained from the physical axes indexes;

– the **OPEN** axes indexes are used while building the communication packet between SMP+ and PC. They can refer to both DSAs, so they are similar to the RAW axes belonging to all DSAs (except the axes which are not set to work in Open modality).

Switch **from logical axes to RAW axes and viceversa**:

– **BLIP/RAAZ packet -** RAW axes --> physical axes --> logical axes --> OPEN axes **- packet to PC**

– **packet from PC -** OPEN axes --> logical axes --> physical axes --> RAW axes **- BLIP/RAAZ packet**

To use the C4G Open modality, the following variables declarations are needed:

– mask of the axes the Open modality applies to:

```
SS_ENTRY C4GOPEN_JNT_MASK
  GBL_INT
  CAT_MAJOR SSL_ARM
  CAT_MINOR SSL_A_CNFG
  TITLE_MDF STE_C4GOPEN_JNT_MASK
  FIELD SST_ARM_C4GOPEN_JNT_MASK
  ATTRIBUTES PRIVRW
  TITLE Joint arm mask
  DESC
  -- Each bit, in the INTEGER data, represents whether the
  -- corresponding axis for that arm is enabled for the
  -- C4GOPEN modality, or not.
  DEND
  RELEASE 3.1
END
```

– C4G Open modality applied to each axis of the Arm:

```
SS_ENTRY C4GOPEN_MODE
  GBL_AINT SSL_MARKER_NAX_ARM 0 GBZ_INT_VAL
  CAT_MAJOR SSL_ARM
```

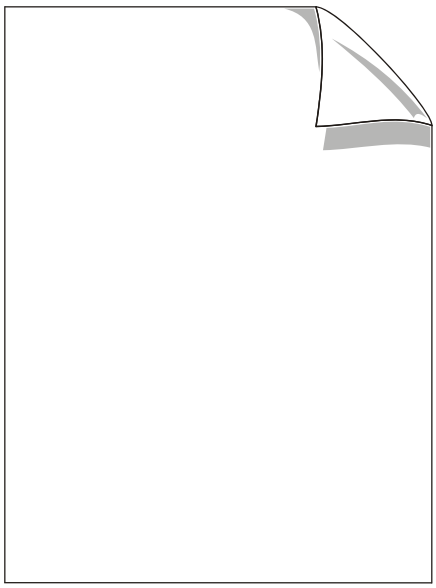
```
  CAT_MINOR SSL_A_CNFG
  TITLE_MDF STE_C4GOPEN_MODE
  FIELD SST_ARM_C4GOPEN_MODE
  ATTRIBUTES PRIVRW
  TITLE C4GOpen modality
  DESC
  -- Each array element contains an integer which identifies
  -- the C4G Open modality applied to that axis.
  DEND
  RELEASE 3.1
END
```

– Enabling bit of the C4G Open modality:

```
SSSW_ARY_HDR  sx_a_along_1d;      /* Arm LONG1 */
```

– C4G Open software option:

```
#define SSB_OPT_C4GOPEN       25 /* C4G OPEN */
#define SSK_OPT_C4GOPEN        (1 << SSB_OPT_C4GOPEN)
```



7. COMMUNICATION

If the system has been properly configured (according to the rules described in Chap.6., not depending on the total amount of axes working in Open modality), in order to better control the communication in a C4G OPEN System, PC must always be active in the communication, answering to the packets sent by SMP+. This restriction is still valid even if at system startup the axes involved in the Open modality are declared to be in "Mode 0" (see par. 14.2 Mode 0 on page 14-2).

- Addresses
- Real time communication
- Communication start
- Regular communication
- Communication end.

7.1 Addresses

The communication between SMP+ and PC is based upon a UDP protocol, where:

- PC acts as a server to address IP 10.2.12.149 (in general: $IPADDRESS_{DSA}$ - 5),
- SMP+ acts as client to address IP 10.2.12.150 (in general: $IPADDRESS_{DSA}$ - 4).

The communication between SMP+ and DSA is based upon a ETH-RAW protocol, where:

- DSA acts as a server to address IP 10.2.12.154 (in general: $IPADDRESS_{DSA}$).

7.2 Real time communication

A **real time communication** must be active between SMP+ and PC. This means a communication mode, meeting the listed below requirements.

Requirements

- PC must send back an answer packet to any packets received from SMP+; if this doesn't happen, the C4G Open modality is disabled, issuing a suitable error message and, depending on the situation, either the C4G System takes the control again or a software error is triggered with a different priority depending on the machine state,
- SMP+ associates a protocol number to the packet (the first field in the packet - see Chap.8. - Communication Packet) and PC must answer back sending a packet in which the first field exactely includes the protocol number provided by SMP+; if this doesn't happen, the system behaviour is the same as described above).

These two restrictions allow both the entire system to work safely (a fault is detected in one interpolator tick), and to software synchronize the communication sender/receiver.

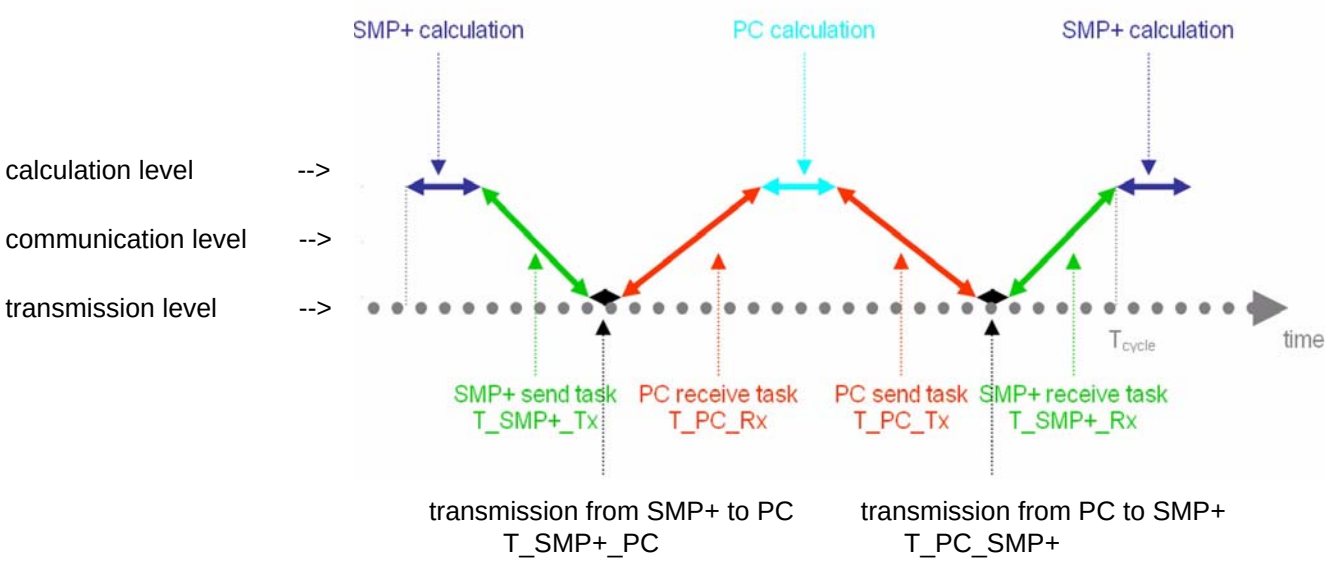
Fig. 7.1 shows the time scheduling of the C4G Open tasks running on both SMP+ and PC. There are three levels:

– **calculation level**: these are processes running on both SMP+ and PC, executing calculation algorithms (trajectory generation, control loops calculations, etc.),

– **communication level**: these are processes running on both SMP+ and PC (belonging to the SMP+ and PC operating systems) handling the packets transmission and reception,

– **transmission level**: electronic signals on the ETH network.

Fig. 7.1 - Communication tasks between SMP+ and PC



It is to be outlined that <u>the real time communication rules must be complied with, at any software/hardware levels!</u>

Especially, it is needed that the calculation algorithms, the transmission/reception tasks, the network interface, the cables and the Switch must comply with the descibed above Requirements, in the expected times. The maximum time interval for PC to answer back to the packet coming from SMP+, is defined in the following Chap.21. - Timing.

Fig. 7.2 - Real Scheduling of processes running on SMP+ and PC simulated by one more SMP+

Fig. 7.2 shows the processes scheduling of a C4G Open operating test. The system is composed by two different SMP+ boards: the first one performs the C4G Open client process, the second one simulates the C4G Open server process which must run on the remote PC.
The communication basic time is 2ms, the used switch and ETH network are not different than the ones which will be chosen to implement the final system.

The processes which handle messages sending and receiving, both on the client and on the server, are referred to as NETT.
The other tasks running on the client, are mainly the ones which handle the trajectory generation.
The OPEN task, on the server, simulates the C4G Open task running on the remote PC.

Note that, on the client, the sending/receiving tasks run twice: this is due to the fact that the client has to communicate with DSA too.

The first two NETT tasks are, respectively, the one sending packets from SMP+ to DSA and the one receiving packets from DSA to SMP+. It is important to outline that packet sending from SMP+ to DSA is started by a deterministic hardware interrupt, whereas the packet reception start also depends on the operations performed by DSA (it is not far from 300us).

The timing window, from the upper diagram to the lower diagram, goes from the moment in which the enquiry sending task ends on the client, to the moment in which the answer sending task ends on the server. As shown in Fig. 7.2, it is less than 100us, complying

with the descibed above Requirements, in the expected times, specified in the following Chap.21. - Timing.

7.3 Communication start

The communication start takes place in three different phases:

– **Phase 1** (PC startup):

a. PC starts before C4G restarts,

b. PC executes a server process opening a UDP socket,

c. PC executes a BIND to the port which is intended for C4G Open communication,

d. PC waits for receiving a packet from SMP+,

– **Phase 2** (C4G startup):

e. C4G starts after saving the OPEN configuration (see Chap.6.),

f. C4G opens a client-side socket to the specified address,

g. C4G sends the first communication packet to PC,

– **Phase 3** (PC first answer):

h. PC receives the first packet sent by SMP+, and sends back a packet whose first field includes the protocol number received from SPM+,

– **Phase 4** (starting the regular communication between SMP+ and C4G):

i. if C4G gets the answer back from PC in the expected times, the two devices are synchronized and the regular communication can start.

7.4 Regular communication

The regular communication requires PC to answer back with the proper protocol number, to each packet sent from SMP+, so that the two devices cannot lose their synchronization.

The other packet fields are fully described in Chap.8. - Communication Packet.

7.5 Communication end

The communication ends when a restart is issued (CCRC command-ConfigureControllerRestartCold).
SMP+ issues the system restart: it no longer requires the synchronization to PC, nor sends the communication packet to it.
PC detects the communication end, as soon as it does not receive the packet from SMP+.
Obviously if, before restarting, the system has not been re-configured (see axes mask in Chap.6. - System initial configuration), C4G restarts in OPEN modality, so it needs the PC presence to properly work.

8. COMMUNICATION PACKET

The current chapter, lists and describes the communication packets **fields**, related to SMP+ and PC communication in OPEN modality.

The content of some fields can be void, depending on the working Modalities; Some restrictions are absolutely to be complied with:

– the sequence,

– the fields total amount,

– the different fields structure.

The following topics are now fully described:

– Communication packets structure

– Communication packets virtualization

– Communication packets dimension.

8.1 Communication packets structure

Tab. 8.1 - Communication packets structure

no.	field type	SMP+	PC
NUM	packet LONG field	protocol number	protocol number
INFPAR	packet LONG field	information/commands (*)	information/commands (**)
CMD	packet LONG field	special information (***)	special information (****)
SMax	axis LONG field	axis mode	axis mode
D1ax	axis FLOAT field	target position [motor turns]	target position [motor turns]
D2ax	axis FLOAT field	target velocity [delta motor turns]	target velocity [delta motor turns]
D3ax	axis FLOAT field	actual position [motor turns]	measures [various]
D4ax	axis FLOAT field	actual velocity [delta motor turns]	velocity component [delta motor turns]
D5ax	axis FLOAT field	reference current [Ampere]	current component [Ampere]
EXT1ax	axis FLOAT field	SMP+ first extra-field	PC first extra-field
EXT2ax	axis FLOAT field	SMP+ second extra-field	PC second extra-field
EXT3ax	axis FLOAT field	SMP+ third extra-field	PC third extra-field

(*) - DRIVE ON signal (C4G_STS_MOVING)
(**) - exiting from OPEN modality signal (EXIT_FROM_OPEN)
(***) - specific functionality depending on the selected mode
(****) - not used yet

8.1.1 Communication packets fields

A detailed description follows for each field:

– SMP+ calculates a protocol number for each being sent packet, and inserts it into the packet first field (NUM). As already explained, PC must answer back sending a packet whose first field includes the protocol number sent by SMP+ (range 0x000 .. 0x7FFF).

– INFPAR field includes information exchanged between SMP+ and PC. By means of this field, SMP+ can inform PC if the DRIVEs are ON, and PC can send SMP+ the EXIT_FROM_OPEN command.
This field is unique in a packet but it is referred to a structure which is able to support until four individual ARMs.
To do that (see Fig. 8.1) the 4 bytes integer number, representing it, is split into 4 sub-fields, one for each individual ARM, and can include 256 different values.

Fig. 8.1 - INFPAR field structure

SMP+ tells PC when the DRIVEs are ON for **ARM i**, by setting byte **i** to the C4G_STS_MOVING (5) value.
PC sends to SMP+ the EXIT_FROM_OPEN command for **ARM i**, by setting byte **i** to the C4G_EXIT_FROM_OPEN (0x01) value.

– CMD field can include special information exchanged between SMP+ and PC. This field can be used to communicate to PC which functionality, among the available ones on the remote system, must be really executed in a specified ***OPEN*** mode.
Example - this field is mainly useful when external PC is supposed to supply the manipulator with the trajectory, executing a very complex path.
To achieve this goal, the path is split in several tracks, each of which with a different motion (the first track to be covered with a sinusolidal motion, the second one linear, the third one circular, etc.). The ***OPEN*** mode still remains the same (the target generation by external PC), whereas the type of motion does change (sinusoidal, linear, circular, etc.)
To specify the motion type, the CMD field is used, i.e. via PDL2 language by setting $ARM_DATA[$i_{ARM}$].C4GOPEN_CMD[1] predefined variable, to the numeric value corresponding to the specified being achieved functionality.
This field too is unique in a packet, but it is related to a structure which is able to support until four individual ARMs (similar to the one shown in Fig. 8.1).

– SMax field is supposed to include the operating mode (special too) chosen for the corresponding axis, according to what is described in Chap.14. - Modalities.

– D1ax and D2ax include the position and velocity references (respectively, in motor turns and delta motor turns) calculated by either SMP+ or PC, depending on the axis modality. These fields must always be present in the packet, even if the operating mode does not require PC to generate the trajectory, so these fields are null.

– D3ax and D4ax fields, belonging to the packet sent by SMP+, include the encoder values corresponding to position (motor turns) and velocity (delta motor turns). These fields always include meaningful values, regardless the modality.
The motors actual positions are to be intended as filtered through the calibration

constants for passing from actual motor turns to ideal motor turns (the motor ideal “0” would never match the actual “0”). This means that the position value complies with the one stored in a generic measure moni.log file. The calibration constants are always available either from PDL2 language or Operator Interface on the C4G Controller.
D3ax field, belonging to the packet sent by PC, includes, if the selected mode does require it, the measure value coming from PC. The unit of measure depends on the performed measuring type. There are several different measuring instruments such as accelerometers, additional encoders, vision systems, force/torque sensors, proximity sensors, etc. Obviously, since each measuring requires a different signal processing procedure, there is an open mode for each type of sensor.

– D4ax and D5ax fields, belonging to the packet sent by PC, include, if the selected mode does require it, the calculated values, respectively, of the position loop (delta motor turns) or of the velocity loop (Ampere) corresponding to the developed control on PC. In this case too, such fields must anyway be present in the packet, even if the operating mode does not require them, and their values are null.
– D5ax field, sent by SMP+, contains the reference current including the dynamic model component (Ampere) commanded to the DSP internal current loops.
– EXT1ax, EXT2ax and EXT3ax fields can be used to transfer extra data values from SMP+ to PC and viceversa.

8.2 Communication packets virtualization

From a conceptual point of view, the communication packet between SMP+ and PC can be virtualized by means of a record called ***C4GOpen_Packet*** including the following fields:

Tab. 8.2 - Abstraction of the C4G Open communication packet

C4GOpen_Packet.NUM	packet protocol number
C4GOpen_Packet.INFPAR	information/parameters between SMP+ and PC (for each packet)
C4GOpen_Packet.CMD	special information between SMP+ and PC
C4GOpen_Packet.AX(i).SM	mode (for each axis)
C4GOpen_Packet.AX(i).DATA	useful information (for each axis)
C4GOpen_Packet.AX(i).DATA.G1	subfield including the first useful information (for each axis)
C4GOpen_Packet.AX(i).DATA.G2	subfield including the second useful information (for each axis)
C4GOpen_Packet.AX(i).DATA.G3	subfield including the third useful information (for each axis)
C4GOpen_Packet.AX(i).DATA.G4	subfield including the fourth useful information (for each axis)
C4GOpen_Packet.AX(i).DATA.G5	subfield including the fifth useful information (for each axis)
C4GOpen_Packet.AX(i).EXT	extra information (for each axis)
C4GOpen_Packet.AX(i).EXT.G1	subfield including the first extra information (for each axis)
C4GOpen_Packet.AX(i).EXT.G2	subfield including the second extra information (for each axis)
C4GOpen_Packet.AX(i).EXT.G3	subfield including the third extra information (for each axis)

From the point of view of the implementation:

```
typedef struct data_ax
```

```
{
    long  SM; // mode
    float D1; // useful data 1
    float D2; // useful data 2
    float D3; // useful data 3
    float D4; // useful data 4
    float D5; // useful data 5
    float EXT1; // extra data 1
    float EXT2; // extra data 2
    float EXT3; // extra data 3
} MVSW_DATA_AX;
#define MVZ_DATA_AX sizeof(MVSW_DATA_AX)
typedef struct data
{
    long NUM; // packet number
    long INFPAR; // DRIVE ON and EXIT FROM OPEN
    long CMD; // special information
    MVSW_DATA_AX AX[2*GBM_NAX_ARM];//data structure for each axis
} MVSW_DATA_PC;
#define MVZ_DATA_PC sizeof(MVSW_DATA_PC)
```

8.3 Communication packets dimension

Tab. 8.3 - Communication packets dimension (in bytes)

HEADER	header of the UDP packet	42 byte/packet
NUM	LONG field for each packet	4 byte /packet
INFPAR	LONG field for each packet	4 byte/packet
CMD	LONG field for each packet	4 byte/packet
SMax	LONG field for each axis	4 byte/axis
D1ax	FLOAT field for each axis	4 byte/axis
D2ax	FLOAT field for each axis	4 byte/axis
D3ax	FLOAT field for each axis	4 byte/axis
D4ax	FLOAT field for each axis	4 byte/axis
D5ax	FLOAT field for each axis	4 byte/axis
EXT1ax	FLOAT field for each axis	4 byte/axis
EXT2ax	FLOAT field for each axis	4 byte/axis
EXT3ax	FLOAT field for each axis	4 byte/axis
TOT	total amount of bytes for each packet	54 byte
	total amount of bytes for each axis	36 byte

9. FROM BLIP/RAAZ TO C4G OPEN PACKET

As alread explained (Chap.6. - System initial configuration), the communication between SMP+ and DSA is based upon exchanging BLIP and RAAZ messages: the first one is created by SMP+ and then sent to DSA; the second one is written by DSA and then received by SMP+.

Chap.8. - Communication Packet includes a detailed description of the fields belonging to the sent and received packets between SMP+ and PC.

The goal of the current chapter is to go deeper in understanding how the two structures coexist in SMP+.

– BLIP
– RAAZ
– Communication between SMP+, PC and DSA.

9.1 BLIP

BLIP is an acronym of the italian words BLocco **I**nformativo **P**eriodico **which means P**eriodical **I**nformational **BL**ock**. It is the packet sent from SMP+ to DSA.**

Tab. 9.1 - BLIP structure

BLIP: command packet towards DSA	
Description (Header)	
1	Packet Identification Code (timeStamp)
1	DSA didn't receive the previous acknowledge
Description (Drive message**)**	
1	Command Code
1	Command Parameter
1	Service Code
5	5 available words, structured as integer and floating, depending on the Service Type
Description (Axis message**)**	
1	Command Identification Code
1	Command Option
2	Current reference position
2	Current reference velocity
2	(DynFF) Velocity feedforward gain
2	(DynFF) Acceleration feedforward gain

Tab. 9.1 - BLIP structure (Continued)

2	(DynFF) Jerk feedforward gain
2	Target position of the current motion
2	Velocity loop extra contribute
2	Current loop extra contribute
1	Percentage limit of the supplied current
1	Type of the used current filter
1	Service request Code
5	5 available words, structured as integer and floating, depending on the Service Type

9.2 RAAZ

RAAZ is an acronym of the italian words Risposta **A**utomatica dell'**Az**ionamento **which means D**rive **A**utomatic **A**nswer**. It is the packet sent back from DSA to SMP+.**

Tab. 9.2 - RAAZ structure

	RAAZ: DSA answer back packet
	Description (Header)
1	Packet Identification Code (timeStamp)
1	DSA has received the previous BLIP
1	Code of the detected error; Zero code is reserved to specify “No Errors”
1	Identification Code of the object which has detected the error condition
1	Positional identifier of the program section which has acknowdleged the error
3	Three suitably structured words to send error parameters
	Description (Drive answer**)**
1	Drive current state
1	Optional parameter of the answer
2	Value of the configurable data
1	Answer Identification Code
3	3 available words, structured as integer and floating, depending on the answer
	Description (Axis answer**)**
1	Axis Status Word
1	Current command (current loop input)
2	Axis current position
2	Axis current velocity
2	Current following error
2	First configurable data
2	Second configurable data
1	Code of the answer to the Service

Tab. 9.2 - RAAZ structure (Continued)

3	3 available words, structured as integer and floating, depending on the answer to the Service.

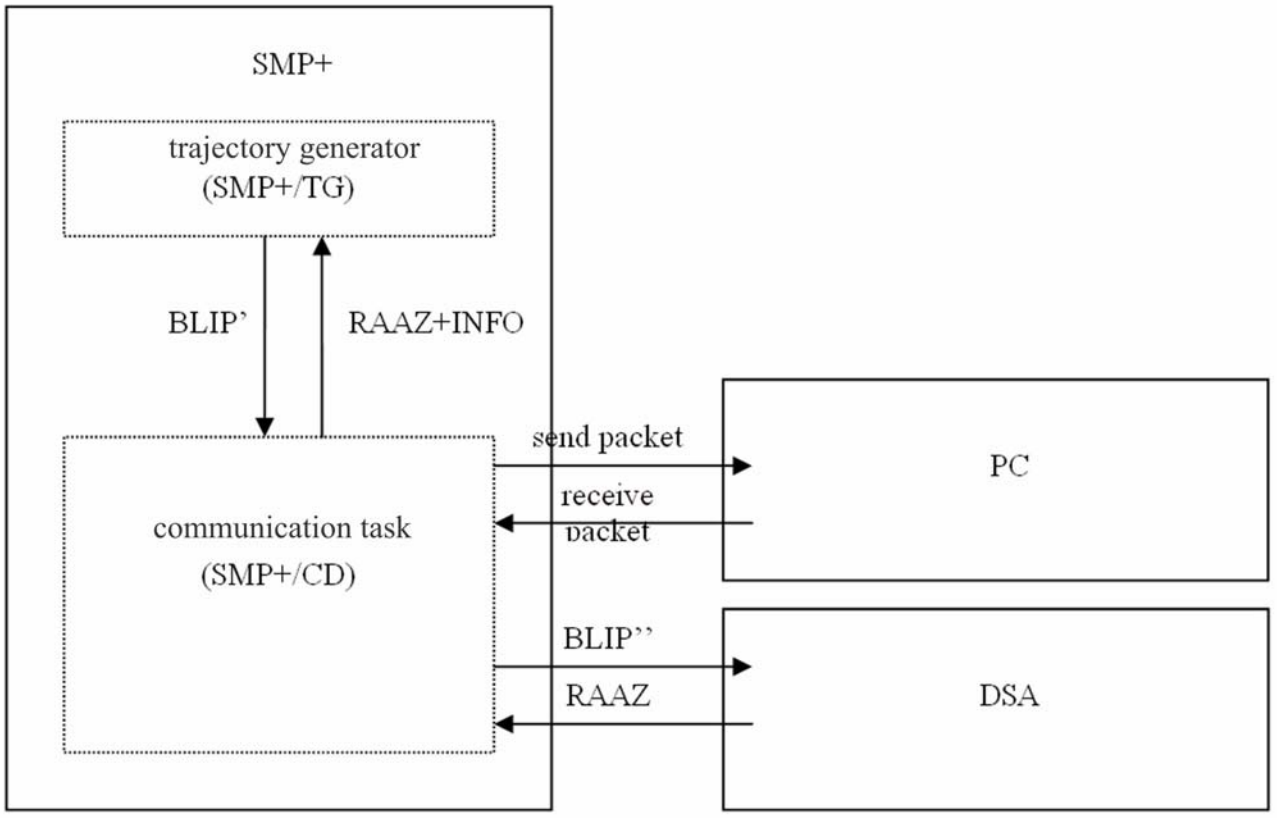
9.3 Communication between SMP+, PC and DSA

Fig. 9.1 shows the communication flow

– based upon C4G Open packets, between SMP+ and PC

– based upon BLIP and RAAZ packets, between SMP+ and DSA

Fig. 9.1 - C4G Open, BLIP and RAAZ packets



The communication timing is as follows (the actor is the SMP+ communication task towards PC - see Fig. 9.1):

a1) receiving packet from PC
a2) receiving BLIP' from the trajectory generator (TG)
a3) sending BLIP'' to DSA

b1) receiving RAAZ from DSA
b2) sending packet to PC
b3) sending RAAZ+INFO to the trajectory generator (TG)

Some more details: as soon as the trajectory generator is ready to send the BLIP to DSA, the communication task gets and modify it, according to the information contained in the packet fields received from PC.

Then, when DSA sends the RAAZ packet to SMP+, it is intercepted by the communication task, in order to modify (adding some information, taken from the BLIP''

it previously received from the trajectory generator) and send it to PC.

Such a RAAZ packet is sent back to the trajectory generator, together with some more information needed by the higher SMP+ software levels.

10. INITIALIZATION PACKET

At the beginning of the communication, SMP+ sends a packet containing some information to allow PC to properly understand the Communication Packet structure.

This phase takes place during the communication initialization procedure; then the synchronization phase between SMP+ and PC is performed.

The information included in the initialization packet is as follows:

– **1st element (long)**: packet protocol number;

– **2nd element (long)**: fields quantity for each packet (i): according to Tab. 8.1, its value is always 3;

– **3rd element (long)**: fields quantity for each axis (j) : according to Tab. 8.1, its value is always 9;

– **4th element (long)**: quantity of axes (k) working in Open modality (max 20: 4 ARM with 2 DSA) (in the example shown in Fig. 10.1, its value is 6);

– **4 ten elements vectors (long)**: they specify the startup Open modality set for each axis (in the example shown in Fig. 10.1 all axes are set to mode 0 at startup);

– **4 ten elements vectors (long)**: list of axes indexes configured to work in Open modality, according to how OPEN axes indexes and logical axes indexes have been mapped (in the example shown in Fig. 10.1 OPEN axes and logical axes directely match);

– **4 ten elements vectors (float)**: calibration constants for each axis;

– **4 ten elements vectors (float)**: max values of reference currents for each axis;

– **4 ten elements vectors (float)**: transmission ratio for each axis;

– **4 ten elements vectors (float)**: kinematic influence coefficients for each axis;

– **4 vettori di 10 elementi (float)**: following error thresholds for each axis;

– **next i elements (long)**: fields meaning for each packet;

– **next j elements (long)**: fields meaning for each axis;

– **next element (long)**: SMP+ sampling period (milliseconds);

– **next element (long)**: software version (major version);

– **next element (long)**: software version (minor version);

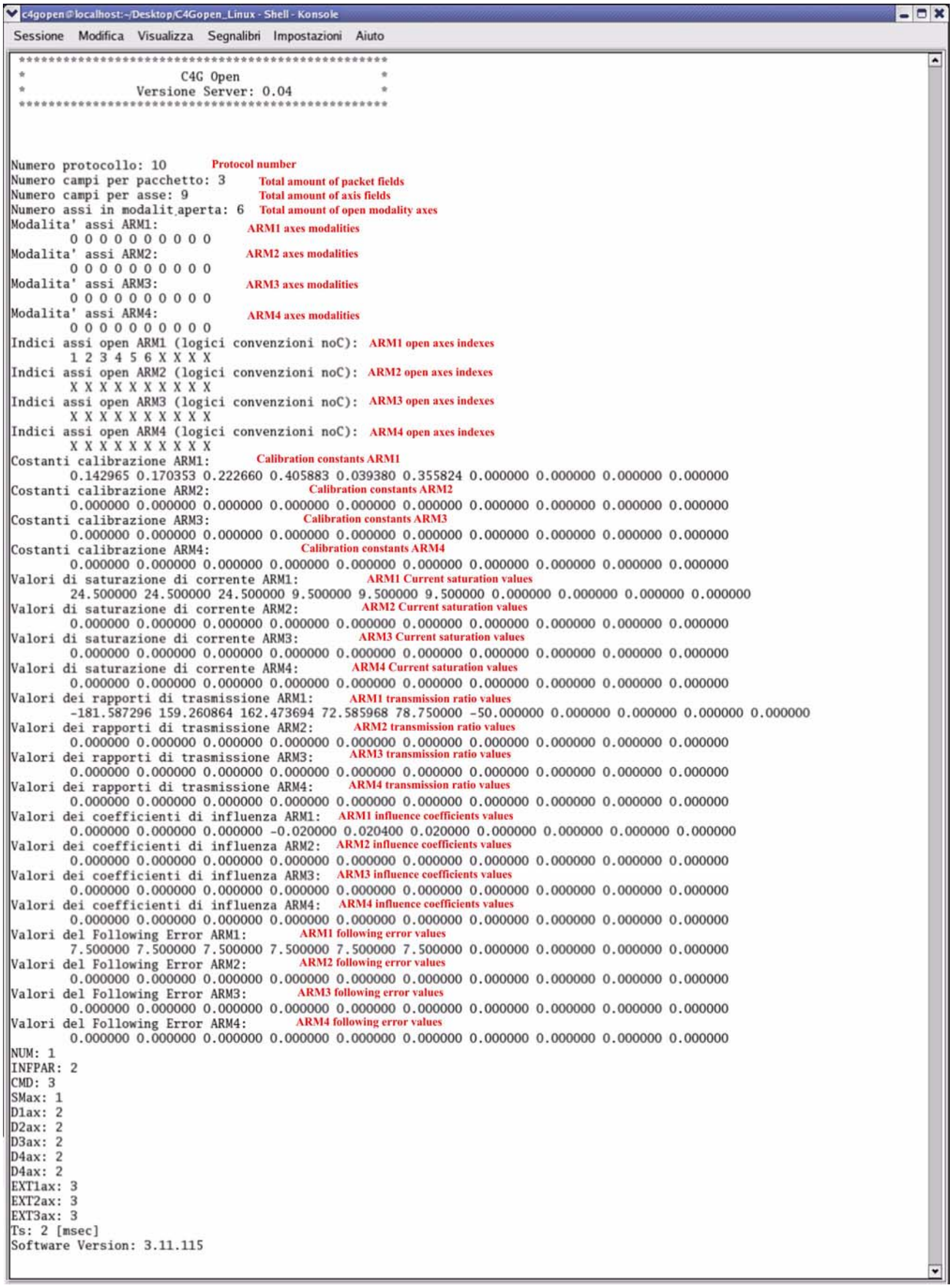
– **next element (long)**: software version (build number).

Listed below the declaration section for the initialization packet:

```
typedef struct syncPck:
{
    long nblip;// protocol number
    long nFieldPck;// packet fields quantity
    long nFieldAx;// axis fields quantity
    long nAxOpen;// open modality axis quantity
    long arm1OpenMask[GBM_NAX_ARM];  // axis specified mode(ARM1)
    long arm2OpenMask[GBM_NAX_ARM];  // axis specified mode (ARM2)
    long arm3OpenMask[GBM_NAX_ARM];  // axis specified mode (ARM3)
    long arm4OpenMask[GBM_NAX_ARM];  // axis specified mode (ARM4)
    long num;// type of the packet 1st field
    long infPar;// type of the packet 2nd field
    long cmd;// type of the packet 3rd field
    long sMax;// type of the axis 1st field
    long d1Ax;// type of the axis 2nd field
    long d2Ax;// type of the axis 3rd field
    long d3Ax;// type of the axis 4th field
    long d4Ax;// type of the axis 5th field
    long d5Ax;// type of the axis 6th field
    long ext1Ax;// type of the axis 7th field
    long ext2Ax;// type of the axis 8th field
    long ext3Ax;// type of the axis 9th field
    long period;// sampling period[msec]
    long arm1OpenAx[GBM_NAX_ARM]; // OPEN axes mask (ARM1)
    long arm2OpenAx[GBM_NAX_ARM]; // OPEN axes mask (ARM2)
    long arm3OpenAx[GBM_NAX_ARM]; // OPEN axes mask (ARM3)
    long arm4OpenAx[GBM_NAX_ARM]; // OPEN axes mask (ARM4)
    float arm1CalData[GBM_NAX_ARM]; // calibration constants(ARM1) [turns]
    float arm2CalData[GBM_NAX_ARM]; // calibration constants (ARM2) [turns]
    float arm3CalData[GBM_NAX_ARM]; // calibration constants (ARM3) [turns]
    float arm4CalData[GBM_NAX_ARM];// calibration constants (ARM4) [turns]
    float arm1AzLim[GBM_NAX_ARM];// max currents (ARM1) [A]
    float arm2AzLim[GBM_NAX_ARM];// max currents (ARM2) [A]
    float arm3AzLim[GBM_NAX_ARM];// max currents (ARM3) [A]
    float arm4AzLim[GBM_NAX_ARM];// max currents (ARM4) [A]
    long majorVersion;// operating system major version
    long minorVersion;// operating system minor version
    long buildNum;// operating system build number
    float arm1AxInfo[GBM_NAX_ARM];// influence coefficients (ARM1) [--]
    float arm2AxInfo[GBM_NAX_ARM];// influence coefficients(ARM2) [--]
    float arm3AxInfo[GBM_NAX_ARM];// influence coefficients(ARM3) [--]
    float arm4AxInfo[GBM_NAX_ARM];// influence coefficients(ARM4) [--]
    float arm1TxRate[GBM_NAX_ARM];// transmission ratios (ARM1) [--]
    float arm2TxRate[GBM_NAX_ARM];// transmission ratios (ARM2) [--]
    float arm3TxRate[GBM_NAX_ARM];// transmission ratios (ARM3) [--]
    float arm4TxRate[GBM_NAX_ARM];// transmission ratios (ARM4) [--]
    float arm1FollowError[GBM_NAX_ARM];// following error thresholds (ARM1) [--]
    float arm2FollowError[GBM_NAX_ARM];// following error thresholds(ARM2) [--]
    float arm3FollowError[GBM_NAX_ARM];// following error thresholds(ARM3) [--]
    float arm4FollowError[GBM_NAX_ARM];// following error thresholds(ARM4) [--]
} MVSW_SYNC_PCK;
#define MVZ_SYNC_PCK sizeof(MVSW_SYNC_PCK)
```

Fig. 10.1shows the initialization packet received from PC. Note that the english information translation is written in red.

Fig. 10.1 - Initialization packet received from PC



```
c4gopen@localhost:~/Desktop/C4Gopen_Linux - Shell - Konsole
Sessione  Modifica  Visualizza  Segnalibri  Impostazioni  Aiuto
************************************************
*                    C4G Open                  *
*              Versione Server: 0.04           *
************************************************



Numero protocollo: 10                    Protocol number
Numero campi per pacchetto: 3            Total amount of packet fields
Numero campi per asse: 9                 Total amount of axis fields
Numero assi in modalit aperta: 6         Total amount of open modality axes
Modalita' assi ARM1:                     ARM1 axes modalities
        0 0 0 0 0 0 0 0 0
Modalita' assi ARM2:                     ARM2 axes modalities
        0 0 0 0 0 0 0 0 0
Modalita' assi ARM3:                     ARM3 axes modalities
        0 0 0 0 0 0 0 0 0
Modalita' assi ARM4:                     ARM4 axes modalities
        0 0 0 0 0 0 0 0 0
Indici assi open ARM1 (logici convenzioni noC):  ARM1 open axes indexes
        1 2 3 4 5 6 X X X X
Indici assi open ARM2 (logici convenzioni noC):  ARM2 open axes indexes
        X X X X X X X X X
Indici assi open ARM3 (logici convenzioni noC):  ARM3 open axes indexes
        X X X X X X X X X
Indici assi open ARM4 (logici convenzioni noC):  ARM4 open axes indexes
        X X X X X X X X X
Costanti calibrazione ARM1:              Calibration constants ARM1
        0.142965 0.170353 0.222660 0.405883 0.039380 0.355824 0.000000 0.000000 0.000000 0.000000
Costanti calibrazione ARM2:              Calibration constants ARM2
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Costanti calibrazione ARM3:              Calibration constants ARM3
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Costanti calibrazione ARM4:              Calibration constants ARM4
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Valori di saturazione di corrente ARM1:          ARM1 Current saturation values
        24.500000 24.500000 24.500000 9.500000 9.500000 9.500000 0.000000 0.000000 0.000000 0.000000
Valori di saturazione di corrente ARM2:          ARM2 Current saturation values
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Valori di saturazione di corrente ARM3:          ARM3 Current saturation values
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Valori di saturazione di corrente ARM4:          ARM4 Current saturation values
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Valori dei rapporti di trasmissione ARM1:        ARM1 transmission ratio values
        -181.587296 159.260864 162.473694 72.585968 78.750000 -50.000000 0.000000 0.000000 0.000000 0.000000
Valori dei rapporti di trasmissione ARM2:        ARM2 transmission ratio values
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Valori dei rapporti di trasmissione ARM3:        ARM3 transmission ratio values
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Valori dei rapporti di trasmissione ARM4:        ARM4 transmission ratio values
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Valori dei coefficienti di influenza ARM1:       ARM1 influence coefficients values
        0.000000 0.000000 0.000000 -0.020000 0.020400 0.020000 0.000000 0.000000 0.000000 0.000000
Valori dei coefficienti di influenza ARM2:       ARM2 influence coefficients values
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Valori dei coefficienti di influenza ARM3:       ARM3 influence coefficients values
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Valori dei coefficienti di influenza ARM4:       ARM4 influence coefficients values
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Valori del Following Error ARM1:         ARM1 following error values
        7.500000 7.500000 7.500000 7.500000 7.500000 0.000000 0.000000 0.000000 0.000000 0.000000
Valori del Following Error ARM2:         ARM2 following error values
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Valori del Following Error ARM3:         ARM3 following error values
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
Valori del Following Error ARM4:         ARM4 following error values
        0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
NUM: 1
INFPAR: 2
CMD: 3
SMax: 1
D1ax: 2
D2ax: 2
D3ax: 2
D4ax: 2
D4ax: 2
EXT1ax: 3
EXT2ax: 3
EXT3ax: 3
Ts: 2 [msec]
Software Version: 3.11.115
```

In order to indicate the specific open mode, for each involved axis, 4 ten elements vectors are used.

Each vector is referred to one ARM (sorted from the first to the fourth) and the vector elements are associated to the 10 logical axes (sorted from the first to the tenth) which can be configured for one ARM.

If the value of the *i*-th element of the *j*-th vector is *k*, this means that logical axis *i* on ARM *j* is configured to work according to the *k* open mode. Furthermore, there are three different types of *k* values, as listed below:

– *k=0* --> the axis does not work in open mode,

– *k=FF* --> the axis is configured to work in OPEN mode, which is not specified at the system startup,

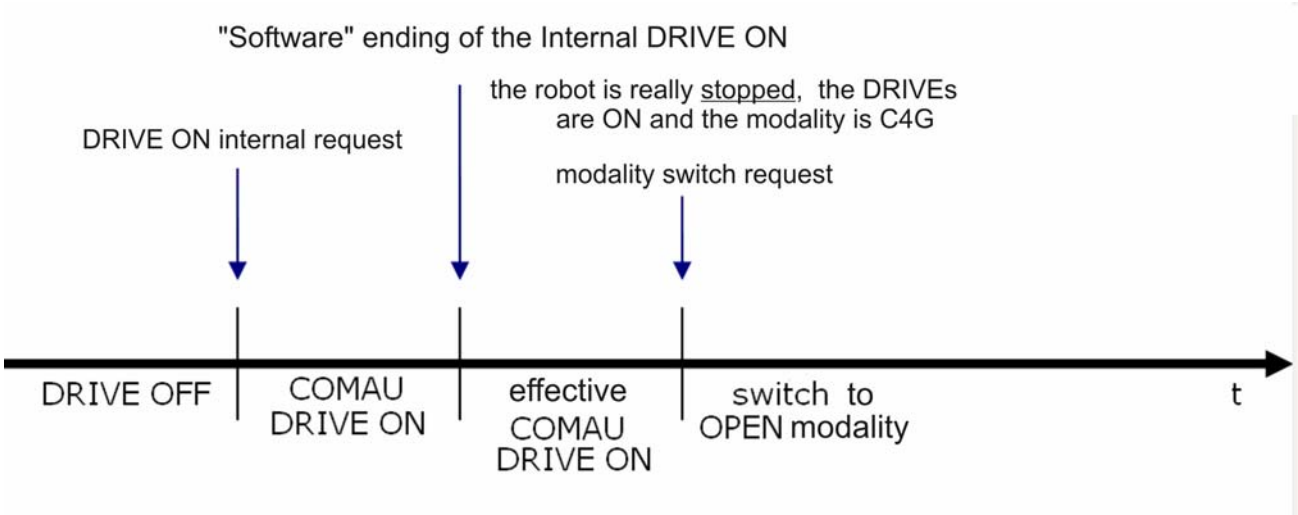
– 0<*k*<FF the axis is configured to work in '*k*' OPEN mode at system startup (it is anyway allowed to change the Open mode later, between a movement and the next one).

The sequence of the *k>0* axes, creates the relationship between logical indexes (for each ARM) and OPEN indexes, as described in Chap.6. - System initial configuration.

11. DRIVE ON PROCEDURE

The machine DRIVE ON procedure (whose timing diagram is shown in Fig. 11.1) is very important.

Fig. 11.1 - DRIVE ON in C4G OPEN system (timing diagram)



At the beginning, system starts in DRIVE OFF state. When DRIVE ON state is required, the activation procedure is divided into two phases.

– First phase - usual DRIVE ON procedure, by means of C4G system: DRIVE ON request, DRIVE ON commands and software controls, motors current activation and, finally, when the machine transitory is finished, the robot is stopped and the DRIVEs are ON, controlled by C4G (SMP+) references and C4G (DSA) drive. During such a phase, the axis mode type must be 0: the system ignores any modality change requests coming both from PC or from PDL2.

– Second phase - when the axis is enabled, the user is allowed to switch to any type of Open modalities. It is up to the external PC, to guarantee that the communication buffer includes, at the same time, both the modality type and any information needed to implement such a modality (e.g. if the PC is supposed to take the control of the position and velocity loops, the suitable current values MUST already be present in the FIRST communication packet, related to the specified modality).

The core of the C4G OPEN philosophy is flexibility, which means being able to dynamically change the specific operating mode (controlled by PC or controlled by SMP+). It is needed to guarantee the servo-mechanism stability during functionality transitions. For this reason, the robot control projects which are supposed to use C4G OPEN software option, must comply the following requirements:

– “simple” (minimum number of control states) position and velocity loops,

– disabling of the forward control (ffv=ffa=ffj=0),

– dynamical disabling of the position loop (Kp=0),

– dynamical disabling of the velocity loop (VelGain=0),

– disabling of the torque loop (TcGain=0),

– disabling of the dynamic model (DynGain=0),

– enabling of the anti-windup systems, both for the position loop and for the velocity loop.

It is to outline that, only during the DRIVE OFF phase, it is allowed to select the group of axes which are enabled to the DRIVE ON.
The command, allowing to enable such an operating modality, is called DRIVE ON DISABLE (DRIVEON_DSBL built-in is used - for further information see **PDL2 Programming Language** manual - **BUILT-IN routines list** chapter).

This functionality can be useful at the beginning of the experimentation, in order to be more concentrated on a limited axes quantity, without interacting with the whole robot, keeping stopped the axes the user is not interested in.

12. SOFT CONTROL SWITCHING

One of the specifically developed technologies for the C4G OPEN system, is ***Soft Control Switching (SCS)***.

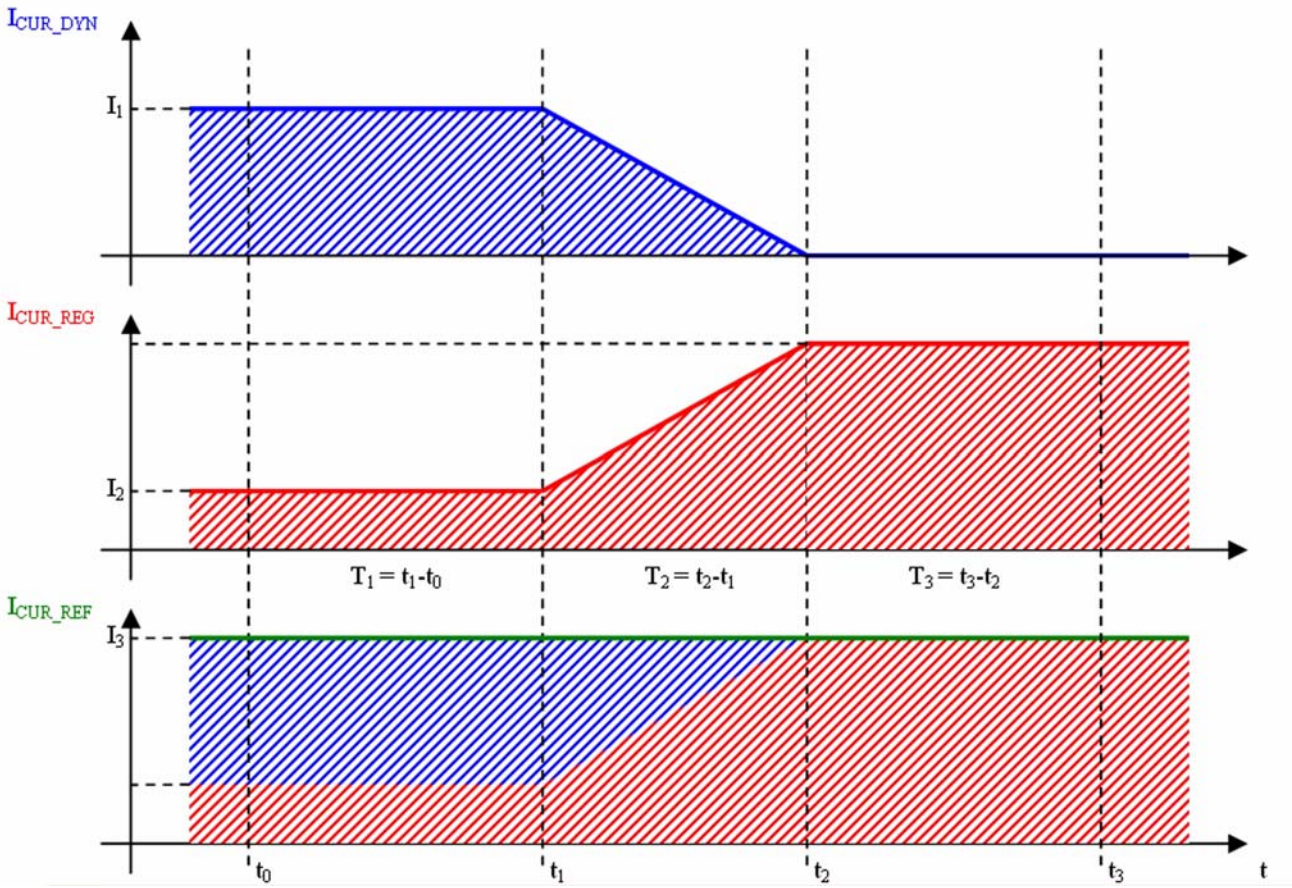
That words are referred to the set of algorithms allowing to gradually switch from a COMAU standard control system, to a control system implemented on external PC.

The first problem is to use the dynamic model “channel” to supply DSA with currents calculated by the PC. To do that, then, at DRIVE ON it is needed to disable the dynamic model, in order to make free use of its “channel”.

To be able to implement such a procedure, a SCS approach is used, as shown in Fig. 12.1, where:

- I_{CUR_DYN} is the current calculated by the dynamic model,
- I_{CUR_REG} is the current calculated by the feedback control,
- I_{CUR_REF} is the reference current for the current loops:
  $I_{CUR\_REF} = I_{CUR\_DYN} + I_{CUR\_REG}$,
- **T1** is called the **pre-switching** time interval and requires the robot not to move and in DRIVE ON state, so that all transitories go over,
- **T2** time interval is the true **switching** phase and it also requires the robot not to move and in DRIVE ON state, to avoid uncontrolled behaviours to come up,
- **T3** is called the **post-switching** time interval and it also requires the robot not to move and in DRIVE ON state, so that all the switching transitories go over too,
- **I1** and **I2** are the total reference currents calculated, respectively, by the dynamic model and the feedback loops,
- **I3** is the reference current which, at the end of the switching phase, is completely evaluated by the feedback control, since the dynamic model has previously been disabled.

Fig. 12.1 - Soft Control Switching: dynamically disabling the dynamic model



T1 and **T3** values must be chosen depending on the robot specific reaction times, whereas **T2** value also depends on the selected switching model.

A simple model is the linear one:

$$\begin{cases} t \in [t_1, t_2] \end{cases} \quad \begin{aligned} \chi(t) &= \frac{t - t_1}{t_2 - t_1} & \chi(t) &\in [0,1] \\ \eta(t) &= \chi(t) & \eta(t) &\in [0,1] \\ \rho(t) &= 1 - \chi(t) & \rho(t) &\in [0,1] \end{aligned}$$

Usign such a model, the dynamic model contribution during the switching phase, can be modulated by means of a variable-gain amplifier:

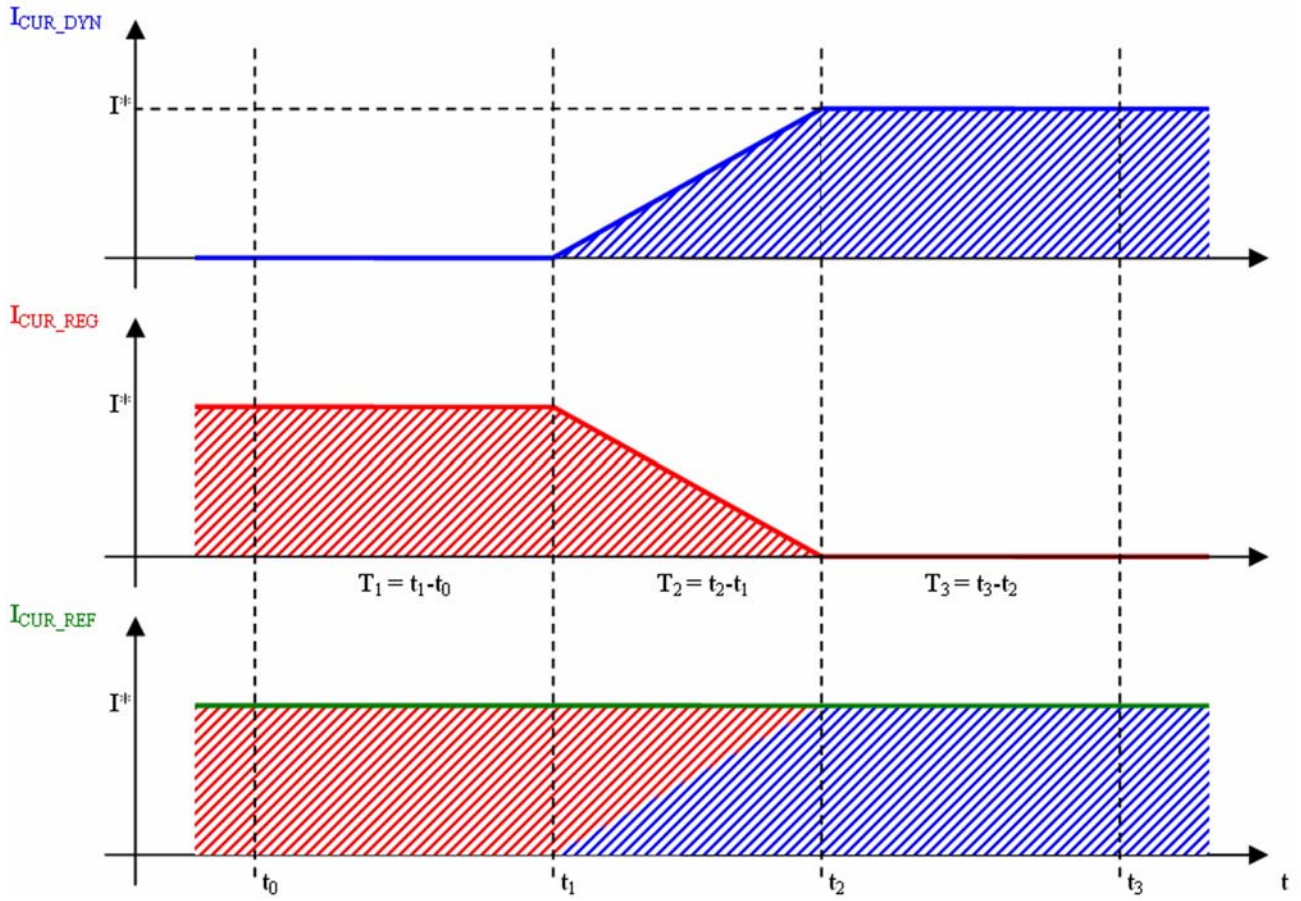
$$t \in [t_1, t_2] \qquad I_{CUR\_DYN}(t) = \rho(t) * I_{CUR\_DYN}(t)$$

I_{CUR_REG} current will consequently adjust itself, since it is the result of feedback computational algorithms.

The SCS technology is also used to switch the C4G Open functionality from a modality in which SMP+ has the complete control of the feedback loops, to the one in which the reference current is controlled by the external PC. Such a current (evaluated by the PC)

is then passed to SMP+ which passes it to DSA, using the dynamic model “channel”.

Fig. 12.2 - Soft Control Switching: switching feedback control systems



According to the principle shown in Fig. 12.2, during the switching phase the dynamic model contribution can be modulated by means of a variable-gain reducer:

$$t \in [t_1, t_2] \qquad I_{CUR\_DYN}(t) = \eta(t) * I_{CUR\_DYN}(t)$$

whereas the control on DSA can be restricted by means of a variable-threshold limiter :

$$t \in [t_1, t_2] \qquad I_{CUR\_REG}(t) = \begin{cases} |I_{CUR\_REG}(t)| \le \rho(t)(I_{max} - I_{min}) & I_{CUR\_REG}(t) \\ |I_{CUR\_REG}(t)| > \rho(t)(I_{max} - I_{min}) & \rho(t)(I_{max} - I_{min})\,\mathrm{sgn}(I_{CUR\_REG}(t)) \end{cases}$$

where:

– I_{max} is the maximum current which can be supplied to the electromechanical system,

– I_{min} is the minimum deliverable current (I_{min}=0).

To perform the inverse switching, from PC complete control, to DSA complete control, it is just needed, during the switching phase, to supply the following currents behaviour:

$$t \in [t_1, t_2] \qquad I_{CUR\_DYN}(t) = \rho(t) * I_{CUR\_DYN}(t)$$

$$t \in [t_1, t_2] \qquad I_{CUR\_REG}(t) = \begin{cases} |I_{CUR\_REG}(t)| \le \eta(t)(I_{max} - I_{min}) & I_{CUR\_REG}(t) \\ |I_{CUR\_REG}(t)| > \eta(t)(I_{max} - I_{min}) & \eta(t)(I_{max} - I_{min})\,\mathrm{sgn}(I_{CUR\_REG}(t)) \end{cases}$$

It is thus obvious that the SCS technology is based upon using a reducer and a limiter which limit both the dynamic model currents and the feedback control currents.

To avoid sharp behaviours of the machine, each control system must be provided with an anti-windup system.
The C4G Controller is already provided with anti-windup systems, wheras it is a constraining requirement to implement feedback loops provided with anti-windup systems on the external PC too.

Fig. 12.3 - Soft Control Switching: block diagram

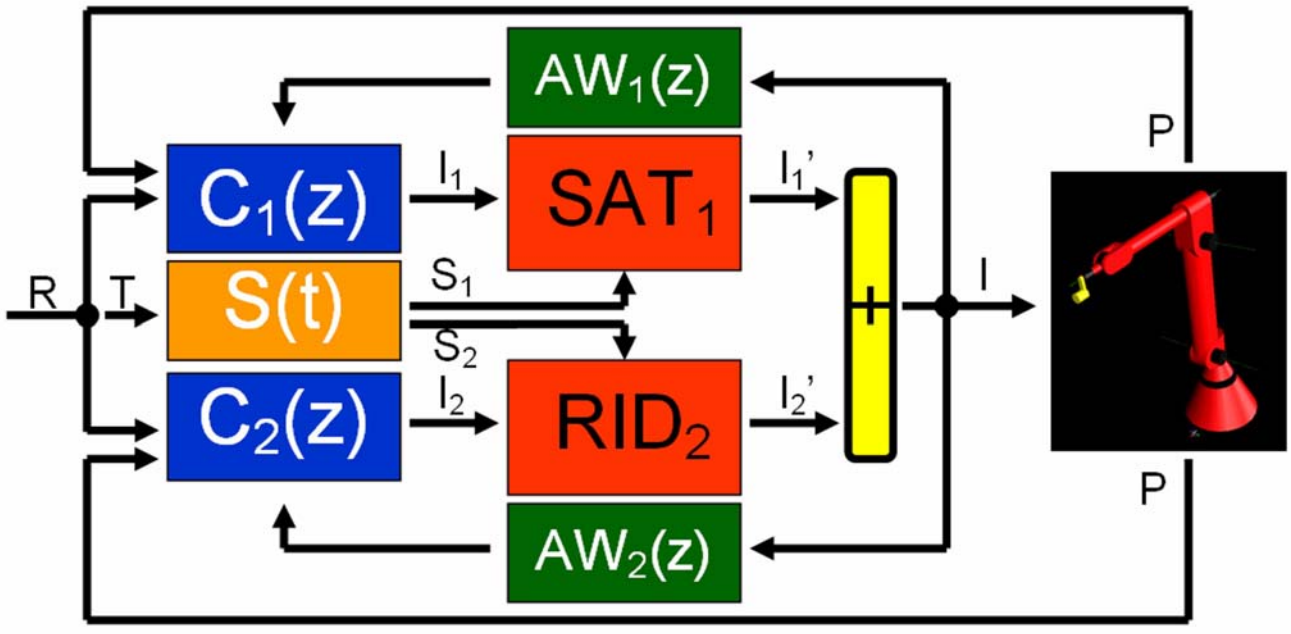


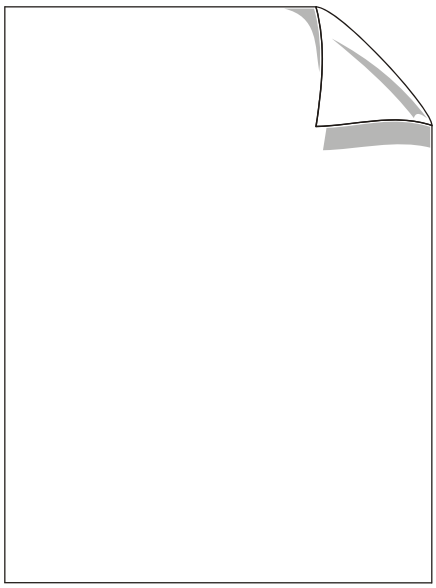
Fig. 12.3 shows the block diagram which implements the SCS technology, where:

- $C_1(z)$ and $C_2(z)$ are referred to the control systems, respectively, of the DSA regulators and the ones implemented on PC,
- SAT_1 is referred to the variable-threshold limiters of the DSA regulators,
- RID_2 is referred to the variable-gain reducers of the variable-gain regulators implemented on PC,
- $AW_1(z)$ and $AW_2(z)$ are referred to the anti-windup systems, respectively, for the DSA regulators and the ones implemented on PC,
- $I_1=I_1(t)$ and $I_2=I_2(t)$ are referred to currents which have been evaluated by, respectively, DSA regulators and PC regulators,
- $I_1'=I_1'(t)$ and $I_2'=I_2'(t)$ are referred to the currents exiting from the limiters, respectively, of the DSA regulators and the ones implemented on PC,
- R=R(t) are the position references to be reached by each robot joint,

- P=P(t) are the joint positions evaluated by the encoders,
- I=I(t)= I_1'(t)+ I_2'(t) are the reference currents for the current loops,
- S (t) is the χ(t) value, calculated by SCS,
- S_1=S_1(t) is the ρ(t) value, calculated by SCS,
- S_2=S_2(t) is the η(t) value, calculated by SCS,
- T=T(t) is the SCS internal timing.

In order to make the most of the SCS technology, it is needed to comply some rules for writing the PDL2 program which controls the C4G Open system:

- **the first motion statement must be preceeded by a DELAY statement lasting $T=T_1+T_2+T_3$ seconds, in order to allow dynamically disabling the dynamic model,**
- **each statement which requires to switch the operating mode from DSA to PC, must be preceeded by a DELAY statement lasting $T=T_1+T_2+T_3$ seconds, to allow the described above transitories to go over.**



13. DYNAMIC HANDLING OF THE OPEN MODALITY

One of the main features of the C4G OPEN functionality, is to be able to dynamically switch the operating mode: it does not require the DRIVE OFF procedure.
The only restriction is that the operating mode switch MUST ONLY be performed between a MOVE statement and the next one.

The implementation of the COMAU C4G Controller "Open" option, allows to create applications in which the standard (SMP+ and MPC) trajectory generation and control processes, interact in a fully flexible mode with additional trajectory generation and control functions, implemented on an external PC.

There are several modalities for the external process to operate on the manipulator; some of the available ones are listed below:

– position additional control (corrections);

– position control;

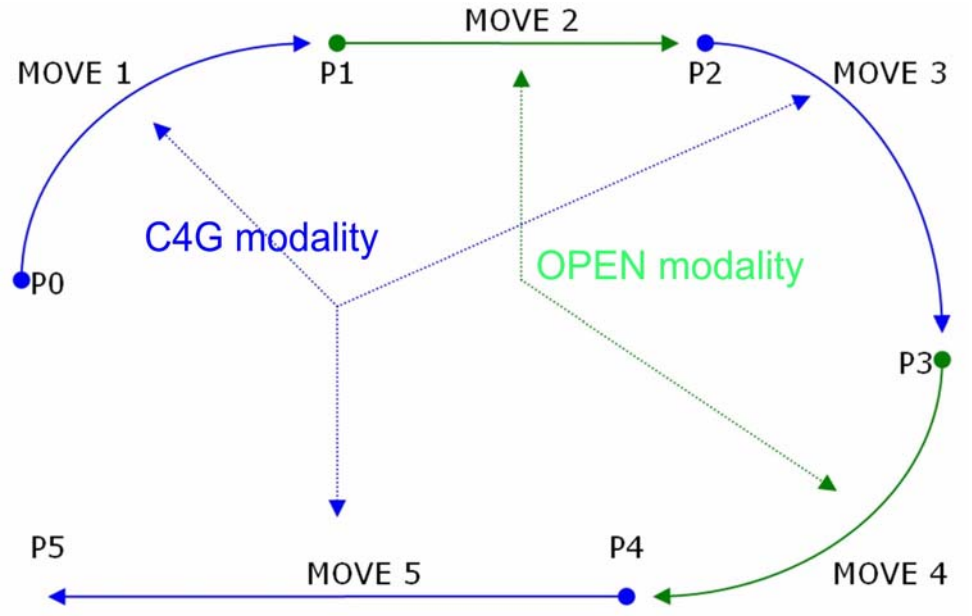
– current additional control;

– trajectory handling;

– modification of the C4G programmed trajectory.

A mixed trajectory example, which points up the possibility of alternating and co-operating in the control actions, between C4G and external PC, is shown in Fig. 13.1.

Fig. 13.1 - Robot motion with different control modes



A plausible PDL2 motion program, implementing the path shown in Fig. 13.1, could be as follows:

```
MOVE TO P1
GO OPEN TYPE A
WAIT FOR PC
MOVE TO P3
GO OPEN TYPE B
MOVE TO P4
WAIT FOR PC
MOVE TO P5
```

At the beginning the robot is in P0 position; then it moves to P1, in usual modality (i.e. fully controlled by C4G), by means of "MOVE TO P1" statement.

When P1 has been reached, the "GO OPEN TYPE A" statement is executed which causes the system to enter the Open modality: SMP+ issues such a command to PC by suitably setting the related field, in the communication packet (see. Chap.8. - Communication Packet).

PC operates in Open modality, properly filling in the packet fields to be sent to SMP+, according to the modalities specified in Chap.14. - Modalities (e.g. providing motors with position references, to move the robot until P2). In the meanwhile, "WAIT FOR PC" statement suspends the PDL2 program execution.

The PDL2 program goes on again as soon as PC, when its operations in open modality have been accomplished, switches the system back to the usual modality, by means of the "EXIT_FROM_OPEN" signal, suitably setting the INFPAR field in the communication packet.

The PDL2 program goes on executing the "MOVE TO P3" statement, in normal modality.

When the robot reaches P3 position, the system switches to OPEN modality again, by means of the "GO OPEN TYPE B" statement.

The PDL2 program does not stop: it executes "MOVE TO P4" (e.g. receiving from PC the motor currents references) and then waits for the PC signal to switch back to the normal modality.

Finally, in normal modality, the robot moves to P5 end position.

Please, note that "GO OPEN TYPE A" and "GO OPEN TYPE B" are very different:

- **the first one does not ask SMP+ to schedule the trajectory while in open modality. So, at the end of this phase, the trajectory is generated by PC only;**
- **"GO OPEN TYPE B" statement, instead, asks SMP+ to schedule the trajectory, together with PC, if needed.**

The OPEN modality end is determined by two different events:

- **SMP+ ends its trajectory, and**
- **PC issues "EXIT_FROM_OPEN" signal.**

This must work not depending on the sequence in which the two above mentioned events occur.

14. MODALITIES

14.1 Introduction

The word **modality** (**mode**) stands for the specific operating type, related to the C4G OPEN functionality.

There are several Open modalities, as listed in the next paragraphs; the difference between them mainly consists of the SMax ("axis mode") value, included in the communication packet exchanged between SMP+ and C4G.

Some modalities are provided to control an axis by means of algorithms developed on external PC, someone else to generate axis trajectories, using a profile calculated by external PC. Then, some modalities are provided too, which use a PC to interface towards an external sensor, in order to get the acquired information to implement a better motion control strategy.

The modality change is communicated to the system by setting (put to 1) the following variable: $ARM_DATA[$i_{ARM}$].C4GOPEN_CMD[2].

Note that such a predefined variable is not reset by the system; if it is to be monitored, it is up to the user to reset it (which means to put it to 0), as soon as it goes to 1.

A detailed description is supplied about the following topics:

- Mode 0
- Mode 0'
- Mode 1
- Mode 2
- Mode 3
- Mode 4
- Mode 5
- Mode 6
- Mode 7
- Mode 8
- Mode 9
- Mode 1xx
- Mode 101 (accelerations control)
- Mode 102 (visual servoing)
- Mode 103 (force control)
- Mode 201
- Mode 202

– Modes summary
– Modes and safety controls

14.2 Mode 0

It is the Open Controller default modality and it is the system standard functionality.

– Functions
– Transmission packets
– Operating steps
– Notes.

14.2.1 Functions

– **SMP+**: trajectory generation, motion control and information exchange with DSA
– **PC**: no meaningful operation

14.2.2 Transmission packets

In the following tables, the REQUIRED not empty packets are written in red; the NOT REQUIRED ones are written in blue.

Tab. 14.1 - Mode 0 communication packet

no.	field type	SMP+	PC
D1ax	single field for each axis	target position [motor turns]	target position [motor turns]
D2ax	single field for each axis	target velocity [delta motor turns]	target velocity [delta motor turns]
D3ax	single field for each axis	actual position [motor turns]	measure [various kinds]
D4ax	single field for each axis	actual velocity [delta motor turns]	velocity contribution [delta motor turns]
D5ax	single field for each axis	reference current [Ampere]	current contribution [Ampere]
E1ax	single field for each axis	dynamic model [Ampere]	
E2ax	single field for each axis	diagonal inertia [Kg*m^2]	

14.2.3 Operating steps

a. SMP+ sends position and velocity references to PC and DSA;
b. DSA evaluates the following error related to the measured position,
c. DSA evaluates the coefficients for the position and velocity loops
d. DSA sends back those information to SMP+
e. SMP+ processes them (stopping the machine if a fault occurs)

f. SMP+ sends the information to PC.

14.2.4 Notes

PC, even if it does not actively participate to the robot motion, must be present and answer back to each packet received from SMP+.

14.3 Mode 0’

Not available to the user: it is reserved to COMAU.

It is a debug modality which allows to check the information exchange between SMP+ and PC.
PC must get the position and velocity references and, without processing them, sends them back to SMP+ again, which finally gives them to DSA.

This allows to check the communication between SMP+ and PC, omitting PC to do any processing else.

– Functions
– Transmission packets
– Operating steps.

14.3.1 Functions

– **SMP+**: trajectory generation, motion control and information exchange with DSA
– **PC**: copies and sends back the information received by SMP+

14.3.2 Transmission packets

Tab. 14.2 - Mode 0’ communication packet

no.	field type	SMP+	PC
D1ax	single field for each axis	target position [motor turns]	target position [motor turns]
D2ax	single field for each axis	target velocity [delta motor turns]	target velocity [delta motor turns]
D3ax	single field for each axis	actual position [motor turns]	measure [various kinds]
D4ax	single field for each axis	actual velocity [delta motor turns]	velocity contribution [delta motor turns]
D5ax	single field for each axis	reference current [Ampere]	current contribution [Ampere]
E1ax	single field for each axis	dynamic model [Ampere]	
E2ax	single field for each axis	diagonal inertia [Kg*m^2]	

14.3.3 Operating steps

a. SMP+ calculates position and velocity references

b. SMP+ sends them to PC

c. PC sends the received information, back to SMP+

d. SMP+ provides DSA with information received from PC.

14.4 Mode 1

It is the most complete open modality: it allows to give the external PC full control.

– Functions
– Transmission packets
– Operating steps
– Notes.

14.4.1 Functions

– **SMP+**: information exchanging with DSA
– **PC**: trajectory generation and motion control

14.4.2 Transmission packets

Tab. 14.3 - Mode 1 communication packet

no.	field type	SMP+	PC
D1ax	single field for each axis	target position [motor turns]	target position [motor turns]
D2ax	single field for each axis	target velocity [delta motor turns]	target velocity [delta motor turns]
D3ax	single field for each axis	actual position [motor turns]	measure [various kinds]
D4ax	single field for each axis	actual velocity [delta motor turns]	velocity contribution [delta motor turns]
D5ax	single field for each axis	reference current [Ampere]	current contribution [Ampere]
E1ax	single field for each axis	dynamic model [Ampere]	
E2ax	single field for each axis	diagonal inertia [Kg*m^2]	

14.4.3 Operating steps

a. PC sends position and velocity references to SMP+

b. SMP+ sends them to DSA;

c. PC calculates the current and/or velocity reference

d. PC send the result to SMP+

e. SMP+ sends it to DSA

f. DSA calculates the following error based upon the measured position

g. DSA sends information back to SMP+

h. SMP+ processes them (stopping the machine if a fault occurs)

i. SMP+ sends them to PC.

14.4.4 Notes

– **It is needed, for safety reasons, that there is the following error check and the saturation of the maximum currents value at PC-side**;

– to exit such a modality, the *mode 1'* (11) indication can be used as signal; the number in brackets is the implementation corresponding code, to be used in PDL2 programs to refer to this modality. See Chap.24. - State machine too.

14.5 Mode 2

It is the modality allowing to evaluate the coefficients for the control loops on the external PC, while SMP+ takes care of the trajectory generation.

– Functions

– Transmission packets

– Operating steps

– Notes.

14.5.1 Functions

– **SMP+**: trajectory generation and information exchanging with DSA

– **PC**: motion control

14.5.2 Transmission packets

Tab. 14.4 - Mode 2 communication packet

no.	field type	SMP+	PC
D1ax	single field for each axis	target position [motor turns]	target position [motor turns]
D2ax	single field for each axis	target velocity [delta motor turns]	target velocity [delta motor turns]
D3ax	single field for each axis	actual position [motor turns]	measure [various kinds]
D4ax	single field for each axis	actual velocity [delta motor turns]	velocity contribution [delta motor turns]
D5ax	single field for each axis	reference current [Ampere]	current contribution [Ampere]
E1ax	single field for each axis	dynamic model [Ampere]	
E2ax	single field for each axis	diagonal inertia [Kg*m^2]	

14.5.3 Operating steps

a. SMP+ sends position and velocity references to both PC and DSA;

b. PC calculates the current and/on velocity reference

c. PC sends the result to SMP+

d. SMP+ sends it to DSA

e. DSA calculates the following error based upon the measured position

f. DSA sends information to SMP+

g. SMP+ processes them (stopping the machine if a fault occurs)

h. SMP+ sends them to PC.

14.5.4 Notes

– **It is needed, for safety reasons, that there is the following error check and the saturation of the maximum currents value PC-side**;

– to exit such a modality, the *mode 2'* (12) indication can be used as signal; the number in brackets is the implementation corresponding code, to be used in PDL2 programs to refer to this modality. See Chap.24. - State machine too.

14.6 Mode 3

It is the modality allowing to control the robot in velocity, from the external PC, while SMP+ takes care of the trajectory generation in velocity only.

– Functions

– Transmission packets

– Operating steps

– Notes.

14.6.1 Functions

– **SMP+**: trajectory generation in velocity only, and information exchanging with DSA

– **PC**: velocity control

14.6.2 Transmission packets

Tab. 14.5 - Mode 3 communication packet

no.	field type	SMP+	PC
D1ax	single field for each axis	target position [motor turns]	target position [motor turns]
D2ax	single field for each axis	target velocity [delta motor turns]	target velocity [delta motor turns]
D3ax	single field for each axis	actual position [motor turns]	measure [various kinds]

Tab. 14.5 - Mode 3 communication packet (Continued)

no.	field type	SMP+	PC
D4ax	single field for each axis	actual velocity [delta motor turns]	velocity contribution [delta motor turns]
D5ax	single field for each axis	reference current [Ampere]	current contribution [Ampere]
E1ax	single field for each axis	dynamic model [Ampere]	
E2ax	single field for each axis	diagonal inertia [Kg*m^2]	

14.6.3 Operating steps

a. SMP+ sends the velocity references to both PC and DSA

b. PC sends the current contribution to SMP+

c. SMP+ sends it to DSA

d. DSA measures the encoders position

e. DSA sends the information back to SMP+

f. SMP+ processes (stopping the machine if a fault occurs) and differentiates hem

g. SMP+ sends the resulting velocities to PC.

14.6.4 Notes

This modality is not currently available. It is expected in future releases.

14.7 Mode 4

It is the modality allowing PC to directely generate the position referenced trajectory.

– Functions
– Transmission packets
– Operating steps
– Notes.

14.7.1 Functions

– **SMP+**: motion control and information exchanging with DSA
– **PC**: trajectory generation

14.7.2 Transmission packets

Tab. 14.6 - Mode 4 communication packet

no.	field type	SMP+	PC
D1ax	single field for each axis	target position [motor turns]	target position [motor turns]
D2ax	single field for each axis	target velocity [delta motor turns]	target velocity [delta motor turns]
D3ax	single field for each axis	actual position [motor turns]	measure [various kinds]
D4ax	single field for each axis	actual velocity [delta motor turns]	velocity contribution [delta motor turns]
D5ax	single field for each axis	reference current [Ampere]	current contribution [Ampere]
E1ax	single field for each axis	dynamic model [Ampere]	
E2ax	single field for each axis	diagonal inertia [Kg*m^2]	

14.7.3 Operating steps

a. PC sends to SMP+ the velocity and position direct references

b. SMP+ sends them to DSA

c. DSA calculates the following error related to the measured position

d. DSA evaluates the coefficients for the position and velocity loops

e. DSA sends this information back to SMP+

f. SMP+ processes them (stopping the machine if a fault occurs)

g. SMP+ sends them to PC.

14.7.4 Notes

– It is suggested that, for safety reasons, there is one more following error control at PC level too, in order to have a fault redundant management;

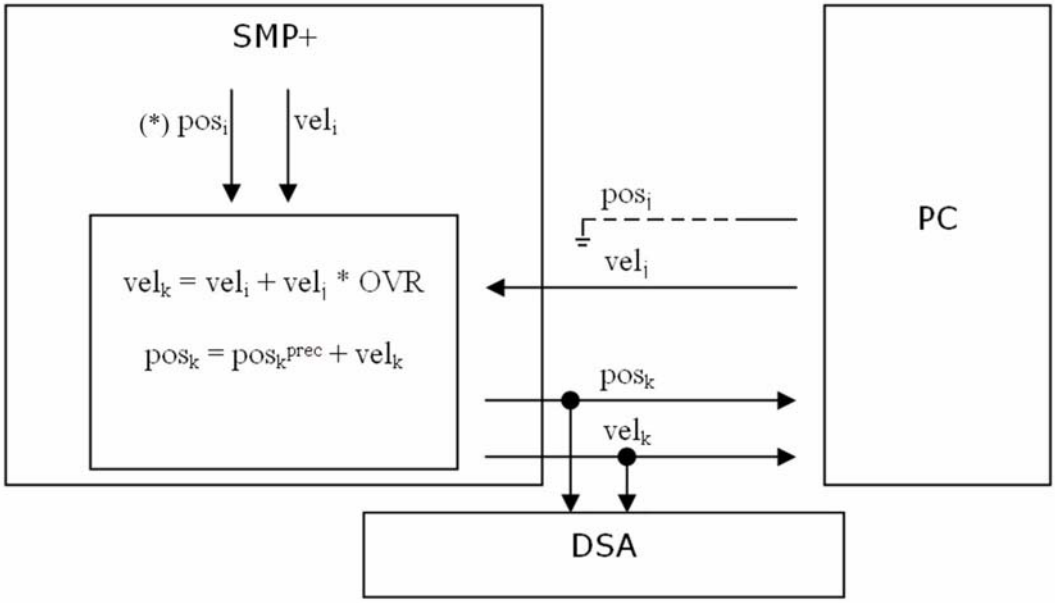
– to exit such a modality, the *mode 4'* (14) indication can be used as signal; the number in brackets is the implementation corresponding code, to be used in PDL2 programs to refer to this modality. See Chap.24. - State machine too.

Since the position reference is to be supplied in a direct way, it is needed to:
– take into account the robot calibration constants,
– move the robot to the starting position of the trajectory to be performed in mode 4, before entering this modality.

14.8 Mode 5

It is the modality allowing PC to indirectely generate the position referenced trajectory.

Fig. 14.1 - Generated trajectory as a delta of general override filtered positions



(*) pos_i is only used to calculate the starting values of the pos_k integrator.

Fig. 14.1 shows the functional block diagram, implementing the mode 5 special configuration, based on the relative positions supplied both by PC and SMP+.

The standard C4G trajectory generator schedules new **pos_i** reference positions and **vel_i** reference velocities. PC calculates the new reference **pos_j** position and **vel_j** velocity. The **pos_k** global reference positions to be sent to DSA, don't take in account the reference positions coming both from C4G trajectory generator and from PC, rather they are calculated related to both the previous step global reference positions and the **vel_k** global reference velocities.

The **vel_k** velocities are calculated adding the **vel_i** reference velocities coming from the trajectory generator to the **vel_j** reference velocities coming from PC, scaled back by the machine general override.

At the end of the movement, i.e. exiting from the open modality, the standard C4G trajectory generator updates **pos_i** setting it to the accumulated **pos_k** value, and clears **pos_k**. Since such a moment, the **pos_i** trajectory sent to DSA will take into account the values calculated by the PC and accumulated in the past, during the trajectory generation in open modality.

WARNING!
There are emergency or anomalous conditions which could cause the system to exit from the open modality without the proper exiting conditions.
It could happen that pos_i cannot be updated to the current value of pos_k. In such situations the robot could perform some sudden and unespected movements owing to the gap existing between the current values of pos_i and pos_k.
BE EXTREMELY CAREFUL!

The required operations to bring the system to work properly again, are as follows:

a. stop executing any test/user program in the current C4G Open modality;

b. **set system variable $C4G_PA:=1;**

c. **reset the modality change system variables**
($ARM_DATA[idx_arm].C4GOPEN_MODE[idx_axis] := 0
which sets the corresponding axes to the default modality);

d. **issue Configure Save All (CSA command);**

e. **issue a Cold Restart (CCRC command);**

f. **if needed, move the robot to the idle position, BEING EXTREMELY CAREFUL!**

Note that steps b., c. and d. are not needed if the user has not saved (CSA command) any values different from the default ones, for such a system variable.

– Functions
– Transmission packets
– Operating steps
– Notes.

14.8.1 Functions

– **SMP+**: motion control and information exchanging with DSA
– **PC**: trajectory generation

14.8.2 Transmission packets

Tab. 14.7 - Mode 5 communication packet

no.	field type	SMP+	PC
D1ax	single field for each axis	target position [motor turns]	target position [motor turns]
D2ax	single field for each axis	target velocity [delta motor turns]	target velocity [delta motor turns]
D3ax	single field for each axis	actual position [motor turns]	measure [various kinds]
D4ax	single field for each axis	actual velocity [delta motor turns]	velocity contribution [delta motor turns]

Tab. 14.7 - Mode 5 communication packet (Continued)

no.	field type	SMP+	PC
D5ax	single field for each axis	reference current [Ampere]	current contribution [Ampere]
E1ax	single field for each axis	dynamic model [Ampere]	
E2ax	single field for each axis	diagonal inertia [Kg*m^2]	

14.8.3 Operating steps

a. PC sends to SMP+ velocity and relative position references

b. SMP+ transforms them into absolute positions

c. SMP+ sends the information to DSA

d. DSA evaluates the following error related to the measured position

e. DSA evaluates the coefficients for the position and velocity loops

f. DSA sends the information back to SMP+

g. SMP+ processes them (stopping the machine if a fault occurs)

h. SMP+ sends them to PC.

14.8.4 Notes

– It is suggested that, for safety reasons, there is one more following error control at PC level too, in order to have a fault redundant management;

– to exit such a modality, the *mode 5'* (15) indication can be used as signal; the number in brackets is the implementation corresponding code, to be used in PDL2 programs to refer to this modality. See Chap.24. - State machine too.

14.9 Mode 6

It is the modality allowing PC to generate the velocity referenced trajectory.

– Functions

– Transmission packets

– Operating steps

– Notes.

14.9.1 Functions

– **SMP+**: motion control and information exchanging with DSA

– **PC**: velocity referenced trajectory generation

14.9.2 Transmission packets

Tab. 14.8 - Mode 6 communication packet

no.	field type	SMP+	PC
D1ax	single field for each axis	target position [motor turns]	target position [motor turns]
D2ax	single field for each axis	target velocity [delta motor turns]	target velocity [delta motor turns]
D3ax	single field for each axis	actual position [motor turns]	measure [various kinds]
D4ax	single field for each axis	actual velocity [delta motor turns]	velocity contribution [delta motor turns]
D5ax	single field for each axis	reference current [Ampere]	current contribution [Ampere]
E1ax	single field for each axis	dynamic model [Ampere]	
E2ax	single field for each axis	diagonal inertia [Kg*m^2]	

14.9.3 Operating steps

a. PC sends velocity references to SMP+

b. SMP+ sends them to DSA

c. DSA measures the position from the encoders

d. DSA evaluates the coefficients for the velocity loop

e. DSA sends this information back to SMP+

f. SMP+ processes them (stopping the machine if a fault occurs)

g. SMP+ sends them to PC.

14.9.4 Notes

This modality is not currently available. It is expected in future releases.

14.10 Mode 7

This modality is a generalization of Mode 5, so all issues described for such a modality apply to mode 7 as well, besides the ones included into the current section. Please, read carefully the WARNING! note, shown in section Mode 5. The essential difference between mode 5 and mode 7 is that mode 5 doesn't allow SMP+ to schedule a movement while executing the modality.

It is the modality allowing PC to supply the trajectory (generated by SMP+) with an **additional** contribution.

– Functions
– Transmission packets
– Operating steps
– Notes.

14.10.1 Functions

– **SMP+**: trajectory generation, motion control and information exchanging with DSA
– **PC**: calculation of the position additional contribution

14.10.2 Transmission packets

Tab. 14.9 - Mode 7 communication packet

no.	field type	SMP+	PC
D1ax	single field for each axis	target position [motor turns]	target position [motor turns]
D2ax	single field for each axis	target velocity [delta motor turns]	target velocity [delta motor turns]
D3ax	single field for each axis	actual position [motor turns]	measure [various kinds]
D4ax	single field for each axis	actual velocity [delta motor turns]	velocity contribution [delta motor turns]
D5ax	single field for each axis	reference current [Ampere]	current contribution [Ampere]
E1ax	single field for each axis	dynamic model [Ampere]	
E2ax	single field for each axis	diagonal inertia [Kg*m^2]	

14.10.3 Operating steps

a. PC sends the position referenced additional contribution to SMP+
b. SMP+ sends the velocity and position references (together with the additional contribution received by PC) to DSA
c. DSA calculates the following error related to the measured position
d. DSA evaluates the coefficients for the position and velocity loops (the contribution calculated by the PC is an additional input value to the position loop)
e. DSA sends this information back to SMP+
f. SMP+ processes them (stopping the machine if a fault occurs)
g. SMP+ sends them to PC;

14.10.4 Notes

– to exit such a modality, the *mode 7'* (17) indication can be used as signal; the number in brackets is the implementation corresponding code, to be used in PDL2 programs to refer to this modality. See Chap.24. - State machine too.

– this modality is useful when a correction is needed related to information coming from external PC.
  This can be done in two separate phases:
  • while moving along the trajectory: in such a case, both C4G and PC take part to the motion, at the same time
  • in the final positioning: in such a case, the double motion termination on the position is checked, PC handles the motion correction as soon as SMP+ accomplishes the trajectory generation, moreover SMP+ continues generating the trajectory even if PC has already finished its motion correction.

– The position referenced additional contribution supplied by PC, is stored by SMP+ in the current position, at each interpolator tick, according to the block diagram shown in Fig. 14.1; thus, the external PC only supplies the last offset related to the position already updated by SMP+ at the previous step.

14.11 Mode 8

It is the modality allowing to add a velocity contribution calculated by external PC, while SMP+ takes care of the trajectory generation.

– Functions
– Transmission packets
– Operating steps
– Notes.

14.11.1 Functions

– **SMP+**: trajectory generation, motion control and information exchanging with DSA
– **PC**: calculation of the velocity additional contribution

14.11.2 Transmission packets

Tab. 14.10- Mode 8 communication packet

no.	field type	SMP+	PC
D1ax	single field for each axis	target position [motor turns]	target position [motor turns]
D2ax	single field for each axis	target velocity [delta motor turns]	target velocity [delta motor turns]
D3ax	single field for each axis	actual position [motor turns]	measure [various kinds]
D4ax	single field for each axis	actual velocity [delta motor turns]	velocity contribution [delta motor turns]
D5ax	single field for each axis	reference current [Ampere]	current contribution [Ampere]
E1ax	single field for each axis	dynamic model [Ampere]	
E2ax	single field for each axis	diagonal inertia [Kg*m^2]	

14.11.3 Operating steps

a. PC sends the additional velocity to SMP+

b. SMP+ sends the position and velocity references (together with the additional contribution received by PC) to DSA

c. DSA calculates the following error related to the measured positon

d. DSA evaluates the coefficients for the position and velocity loops (the contribution calculated by PC is an additional input value to the velocity loop)

e. DSA sends this information back to SMP+

f. SMP+ processes them (stopping the machine if a fault occurs)

g. SMP+ sends them to PC.

14.11.4 Notes

This modality is not currently available. It is expected for future releases.

14.12 Mode 9

It is the modality allowing to add a current contribution calculated by external PC, while SMP+ generates the trajectory.

- Functions
- Transmission packets
- Operating steps
- Notes.

14.12.1 Functions

- **SMP+**: trajectory generation, motion control and information exchanging with DSA
- **PC**: calculation of the current additional contribution

14.12.2 Transmission packets

Tab. 14.11- Mode 9 communication packet

no.	field type	SMP+	PC
D1ax	single field for each axis	target position [motor turns]	target position [motor turns]
D2ax	single field for each axis	target velocity [delta motor turns]	target velocity [delta motor turns]
D3ax	single field for each axis	actual position [motor turns]	measure [various kinds]
D4ax	single field for each axis	actual velocity [delta motor turns]	velocity contribution [delta motor turns]
D5ax	single field for each axis	reference current [Ampere]	current contribution [Ampere]
E1ax	single field for each axis	dynamic model [Ampere]	

Tab. 14.11- Mode 9 communication packet

no.	field type	SMP+	PC
E2ax	single field for each axis	diagonal inertia [Kg*m^2]	

14.12.3 Operating steps

a. PC sends the current additional contribution to SMP+

b. SMP+ sends the position and velocity references, and the current additional contribution (received from PC) to DSA;

c. DSA calculates the following error related to the measured position

d. DSA evaluates the coefficients for the position and velocity loops (the contribution calculated by PC is an additional input value to the velocity loop)

e. DSA sends this information back to SMP+

f. SMP+ processes them (stopping the machine if a fault occurs)

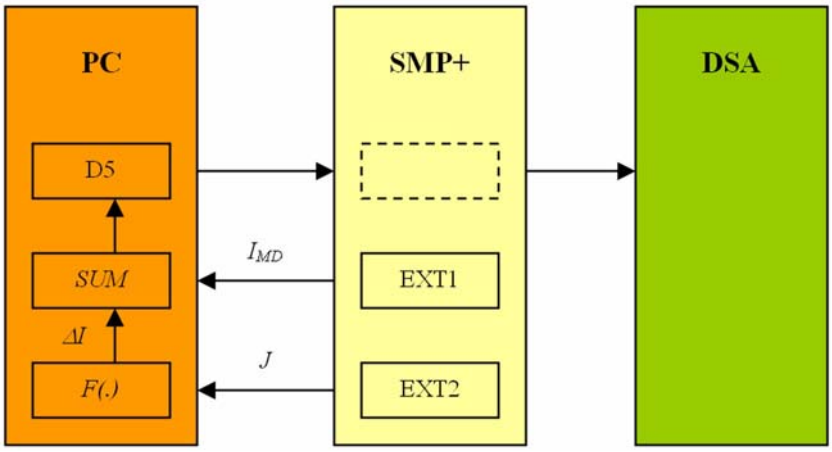
g. SMP+ sends them to PC.

14.12.4 Notes

– In open modality, the dynamic model calculated by SMP+ is no longer sent to DSA, because the “channel” is already used to provide DSA with the PC calculated data.
If, while in this modality, it is needed to use the dynamic model, PC must provide SMP+ with this information too.
The dynamic model current either can be calculated by PC or can go through PC, coming from SMP+.
This information is included in EXT1 field, belonging to the packet from SMP+ to PC (see Tab. 14.11).
EXT2 field includes the diagonal inertia calculated by SMP+; this information too can be used by PC for its own calculations.

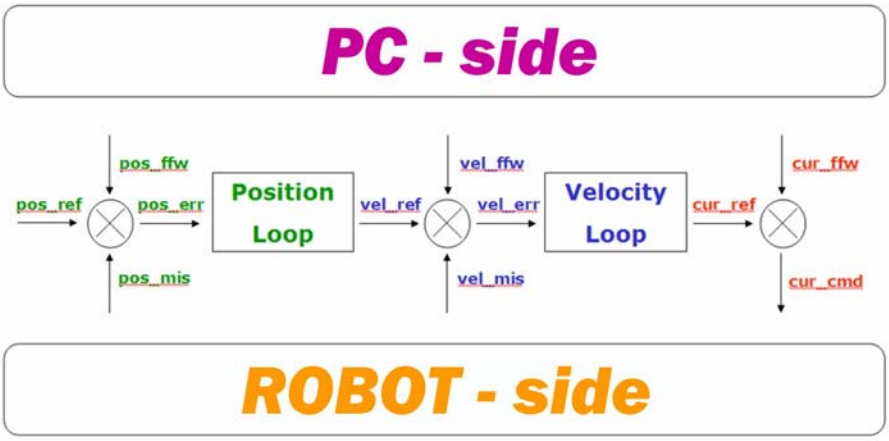
The following block diagram describes such topics (see Fig. 14.2).

Fig. 14.2 - Current additional contribution from PC, using the dynamic model calculated by SMP+.



To be able to discriminate among the additional contributions that remote PC can provide the motion control with, the DSA control blocks internal structure is shown in Fig. 14.3.

Fig. 14.3 - Position, velocity and current additional contribution actions



The position additional contribution, coming from PC (pos_ffw), is input to the Position Loop, together with the position reference calculated by SMP+ (pos_ref) compared with the measured position coming from the encoders (pos_mis).
The same applies to the Velocity Loop whose reference is the Position Loop output (vel_ref).
Finally, the current command to DSA (cur_cmd) comes from the Velocity Loop output (cur_ref) and the current additional contribution calculated by PC (cur_ffw).

14.13 Mode 1xx

It is the modality used to interface C4G to an external sensor.

In such a modality, PC is able to send just one measure to SMP+, but this does not prevent from handling more than one sensor, using PC and then providing SMP+ with

the final processing.

In the C4G OPEN system, a distinction must be made between **physical** connection and **extended control architecture**.

From a “physical” point of view, C4G OPEN functioning is based upon a real-time Ethernet “T” connection, in which there are 3 devices communicating one another:

– SMP+ is the only “client” on the network,

– DSA and PC are the client’s two “servers”.

The external-to-C4G world must pass through PC, which can be seen as the device integrating several sensors, of different kinds, connected in different ways (USB, serial, PCI boards, etc.) and with sometimes different sampling intervals.
Thus, that’s PC which integrates a network that is wide and with several sensors.

This implements the maximum level of flexibility and opening, because the skill of acquiring, interpreting, processing and integrating any information coming from the external sensors, is located on the PC.

Even if at the **physical** connection level, PC is a **server**, from the extended control architecture and **logic** point of view, the system skill is distributed, being located both on SMP+ and on PC. The physical connection is mainly to be seen just as a pure open medium, to drive the motors.

The so described architecture, allows to implement advanced robot control applications.

To better explain it, three examples of control with external sensors applications, are provided in the following sections:

– control with integrated IMU module (inertial navigation) (see par. 14.13.5 Mode 101 (accelerations control) on page 14-19);

– control with integrated vision system (see par. 14.13.6 Mode 102 (visual servoing) on page 14-20);

– control with force sensor (see par. 14.13.7 Mode 103 (force control) on page 14-21).

The following topics are described:

– Functions

– Transmission packets

– Operating steps

– Notes.

14.13.1 Functions

– **SMP+**: trajectory generation, motion control and information exchanging with DSA

– **PC**: external sensor interface.

14.13.2 Transmission packets

Tab. 14.12- Mode 1xx communication packet

no.	field type	SMP+	PC
D1ax	single field for each axis	target position [motor turns]	target position [motor turns]

Tab. 14.12- Mode 1xx communication packet (Continued)

no.	field type	SMP+	PC
D2ax	single field for each axis	target velocity [delta motor turns]	target velocity [delta motor turns]
D3ax	single field for each axis	actual position [motor turns]	measure [various kinds]
D4ax	single field for each axis	actual velocity [delta motor turns]	velocity contribution [delta motor turns]
D5ax	single field for each axis	reference current [Ampere]	current contribution [Ampere]

14.13.3 Operating steps

a. PC supplies SMP+ with the measure coming from the external sensor

b. SMP+ sends velocity and position references to both PC and DSA, taking into account the measure from PC (if needed by the modality)

c. DSA calculates the following error related to the measured position

d. DSA evaluates the coefficients for the position and velocity loops related to the measure coming from PC (if the modality requires it)

e. DSA sends this information back to SMP+

f. SMP+ processes them (stopping the machine if a fault occurs)

g. SMP+ sends them to PC.

14.13.4 Notes

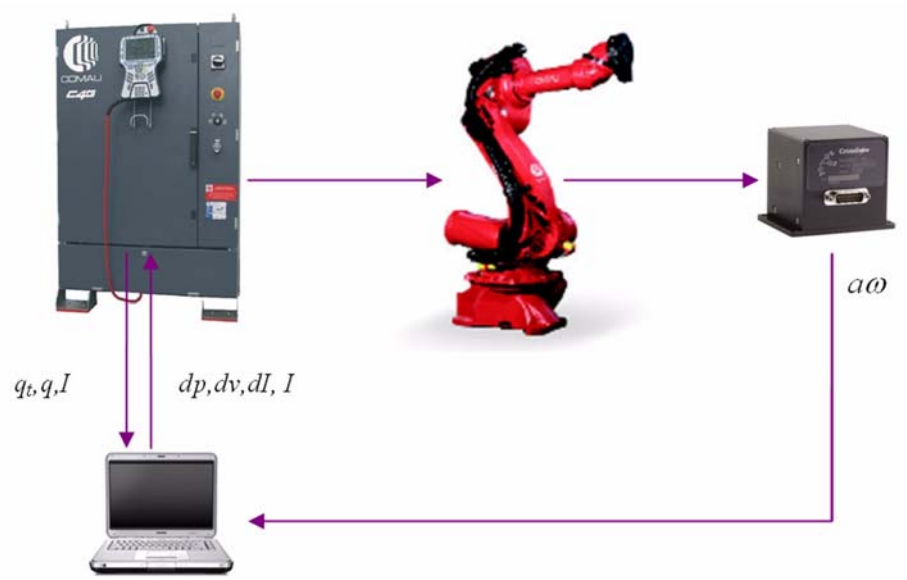
– It is to be outlined that such a measure can be used to generate the trajectory and/or to evaluate the coefficients for the position and velocity loops. To discriminate the different functioning modes, the open modality must be specified, as described in the following paragraphs.

This modality is not currently available. It is expected for future releases.

14.13.5 Mode 101 (accelerations control)

The external sensor is an accelerometer, so the information is handled at control loop level. Fig. 14.4 shows a configuration using an accelerometer.

Fig. 14.4 - Architecture example - C4G OPEN with acceleration sensor.



14.13.6 Mode 102 (visual servoing)

The external sensor is a videocamera, so the information is handled at the trajectory generation level, in order to update the target related to the offset between the position measured by the encoders, and the tool position. Fig. 14.5 and Fig. 14.6 show a C4G OPEN configuration using the external vision sensor.

Fig. 14.5 - Architecture example - C4G OPEN with videocamera (a)

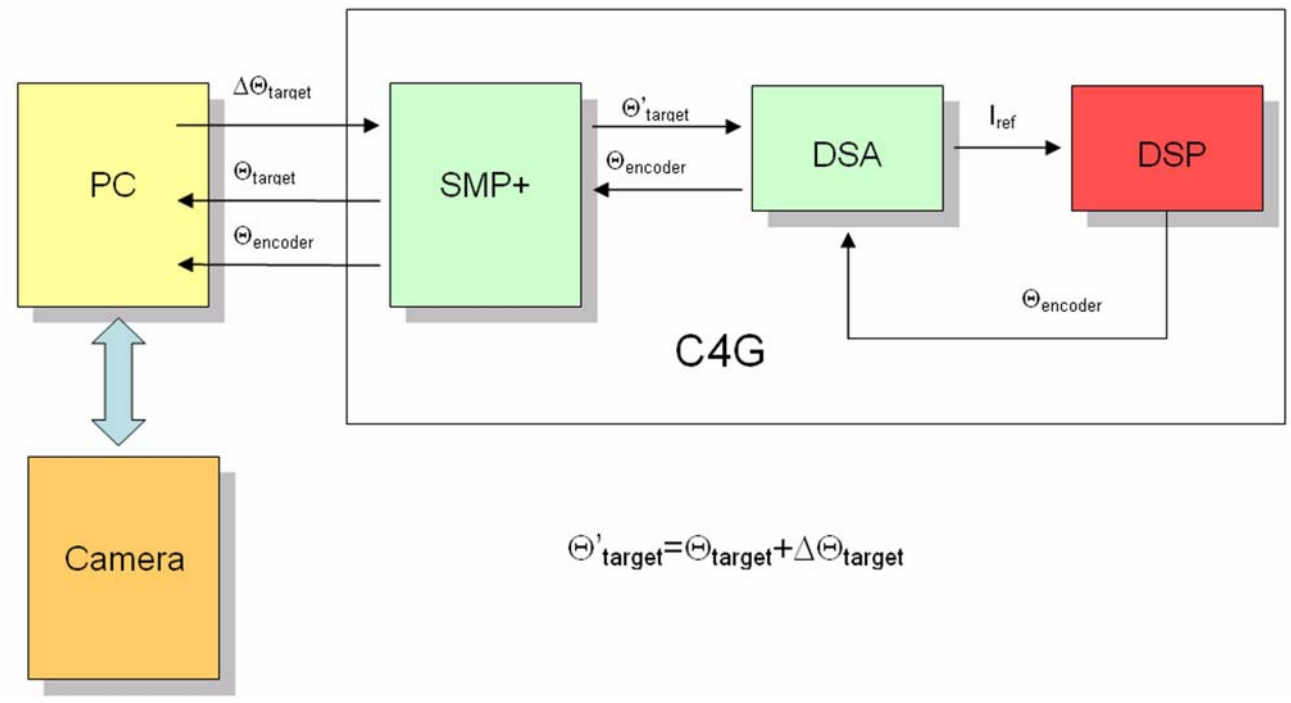
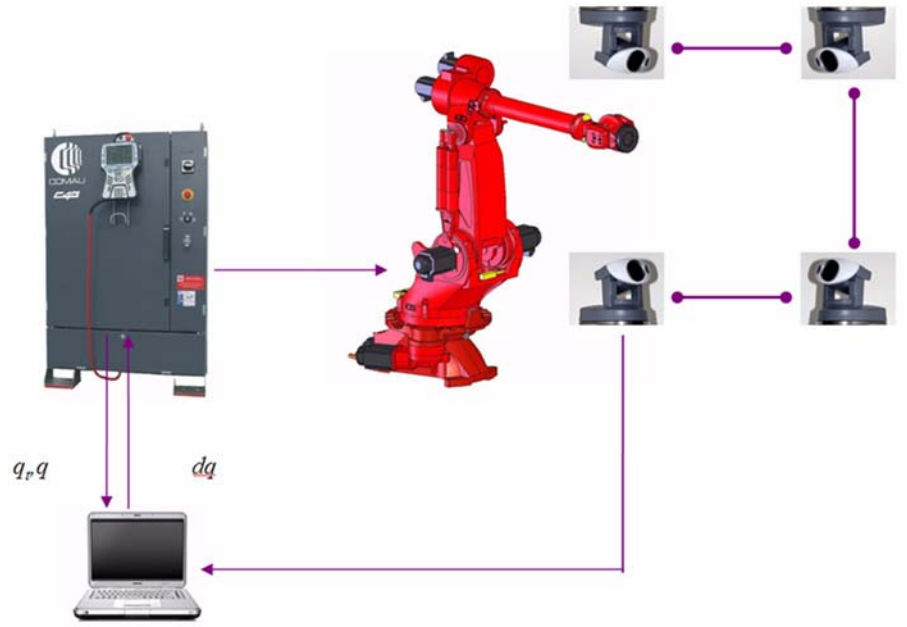


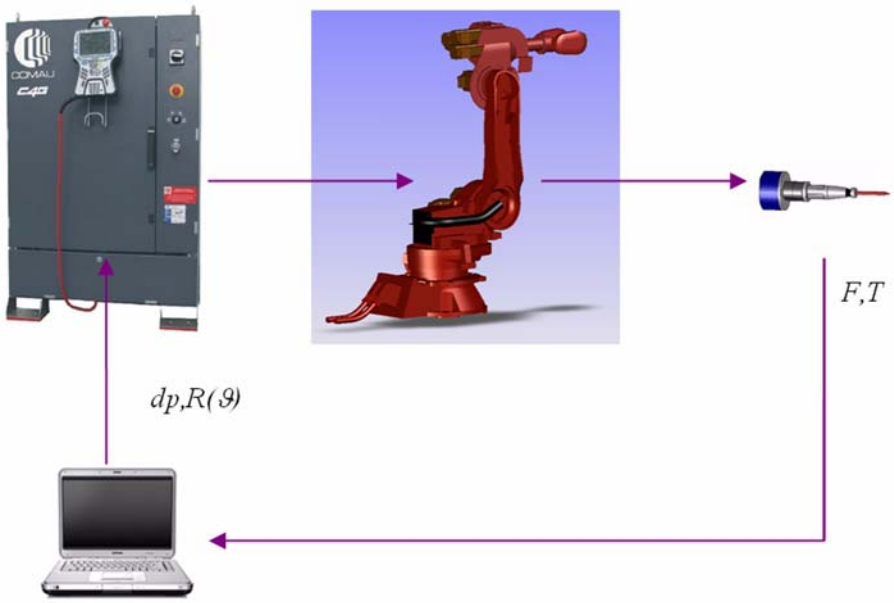
Fig. 14.6 - Architecture example - C4G OPEN with videocamera (b)



14.13.7 Mode 103 (force control)

The external sensor is a force/torque one, so the information is handled at trajectory generation level, to update the target related to the contact force between the robot and the external environment. Fig. 14.7 shows a C4G OPEN configuration using a force/torque sensor.

Fig. 14.7 - Architecture example - C4G OPEN with force sensor



14.14 Mode 201

This modality allows to generate a cartesian trajectory calculated by the external PC. In such a case a complete C4G OPEN revision is needed.

Since now, in fact, PC has been connected to SMP+ and DSA; this meant that PC influence was restricted to either position, velocity or current at joint level.

In this modality, instead, PC is allowed to generate a cartesian trajectory. To implement it, the kinematic inversion is needed, which is included in the SMP+ software.

– Operating steps

– Notes.

14.14.1 Operating steps

a. PC sends the cartesian reference to SMP+

b. SMP+ supplies the upper software levels of the trajectory generation, with such a cartesian reference received from PC

c. SMP+ transforms it in a joint reference

d. SMP+ sends the position and velocity references to DSA

e. DSA calculates the following error related to the measured position

f. DSA evaluates the coefficients for the position and velocity loops

g. DSA sends this information back to SMP+

h. SMP+ processes them (stopping the machine if a fault occurs) and trasforms them into cartesian positions

i. SMP+ sends them to PC.

14.14.2 Notes

– Since the data flow needs to use the trajectory generator software levels, the delay between the imposed reference and its actual execution, is very long.
This is one of the reasons for C4G OPEN to operate in joint world only, possibly performing the joint to cartesian to joint transformations, on remote PC.

See par. 14.15.2 Notes on page 14-23 for further information.

This modality is not currently available. It is expected for future releases.

14.15 Mode 202

This modality allows to supply a position additional contribution, calculated by external PC in cartesian world.

The operating steps are similar to the ones of the previous modality. The only difference is that the reference supplied by PC is not direct: it has to be added to the cartesian reference calculated by SMP+.

– Operating steps

– Notes.

14.15.1 Operating steps

a. PC sends the position additional contribution to SMP+

b. SMP+ supplies the upper software levels of the trajectory generator with such an information, in order to build the absolute cartesian reference and to transform it into a joint reference

c. SMP+ sends position and velocity references to DSA

d. DSA calculates the following error related to the measured position

e. DSA evaluates the coefficients for the position and velocity loops

f. DSA sends this information back to SMP+

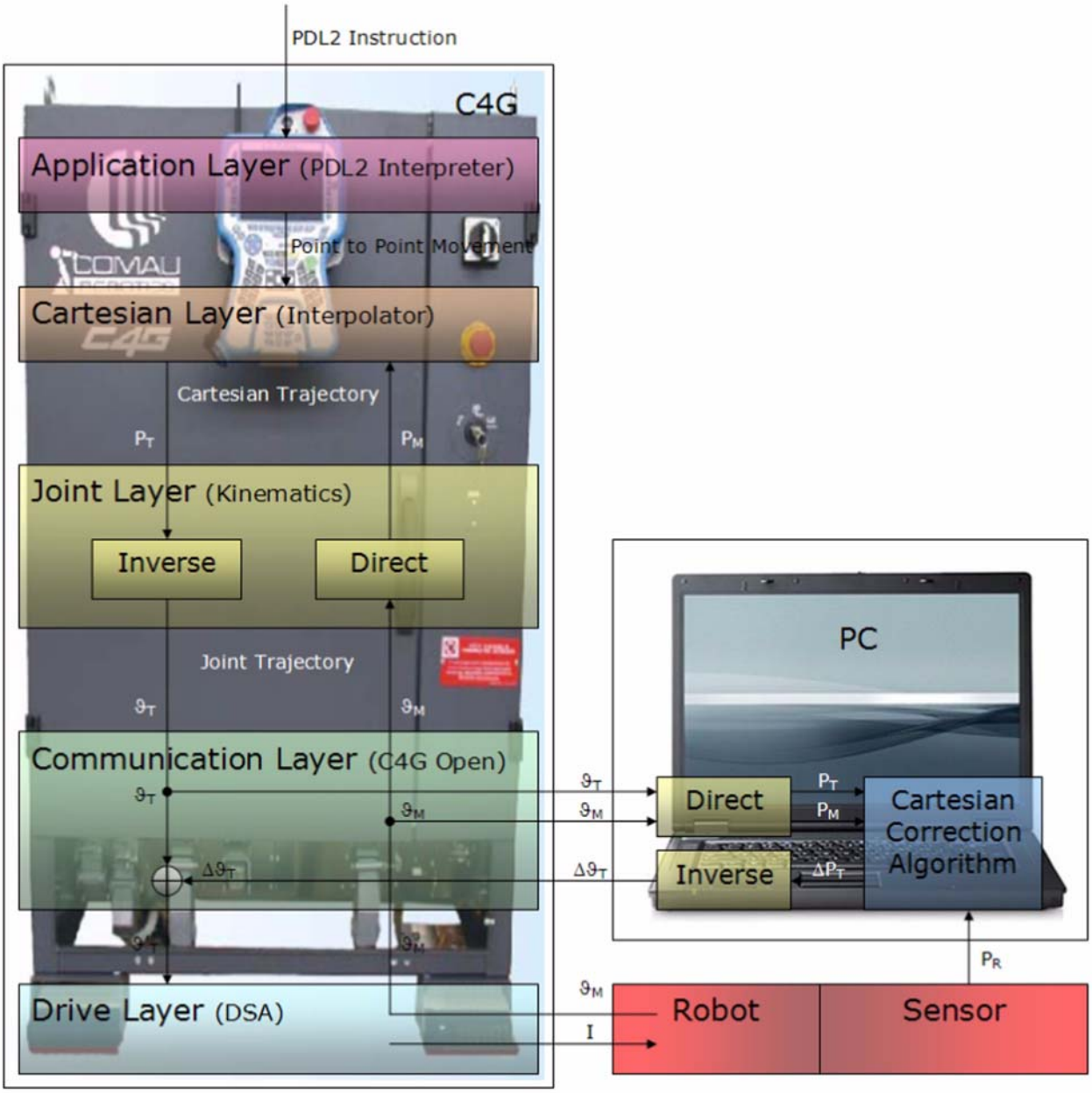
g. SMP+ processes them (stopping the machine if a fault occurs) and trasforms them into cartesian positions

h. SMP+ sends them to PC.

14.15.2 Notes

– Since the data flow needs to use the trajectory generator software levels, the delay between the imposed reference and its actual execution, is very long.
This is one of the reasons for C4G OPEN to operate in joint world only, possibly performing the joint-to-cartesian-to-joint transformations, on remote PC.

  In the below figure a suitable way is shown to use C4G Open technology in Cartesian frame and Joint frame.



The above figure shows all of software levels that manage motion within the C4G architecture:

– "Application Layer": the interpretation software of the PDL2 robot programming language, acts at this level; the layer receives the motion instructions (e.g.: MOVE TO P1 and MOVE TO P2) and gives the motion overall information (e.g.: either Cartesian or Joint motion, P1 position and P2 position, the value of the cruise speed and so on);

– "Cartesian Layer": the software of trajectory generation in Cartesian frame, acts at this level; it receives the movement overall information supplied by the upper level and calculates the real-time Cartesian trajectory at the interpolator sampling rate;

– "Joint Layer": the software of trajectory generation in Joint frame, acts at this level; it receives the Cartesian trajectory supplied by the upper layer and real-time converts it into Joint trajectory at the interpolator sampling rate, the software also

evaluates the inverse kinematics: from the Joint positions measured by encoders it calculates the Cartesian positions using direct kinematics;

– "Communication Layer": the communication software acts at this level; it receives the real-time Joint trajectory supplied by the upper layer, manages the communication between SMP+ and DSA (receiving positions measured by encoders and sending the reference positions) and C4G Open, if enabled, manages the communication between SMP+ and external PC (sending the motor positions and the Joint trajectory and receiving the correction to be applied to the Joint trajectory);

– "Drive Layer": the drive software acts at this level; it receives the final Joint trajectory from the upper layer, evaluates the reference currents to supply the drives with, and sends the motor positions measured by encoders, to the upper level.

The above figure also shows the minimum set of software layers running on external PC, to manage an external sensor in Cartesian frame:

– "Direct": software module that implements direct kinematics (coordinate transformation from Joint frame to Cartesian frame of measured positions and reference positions) exactly like C4G system software does;

– "Inverse": software module that implements inverse kinematics (coordinate transformation from Cartesian frame to Joint frame of correction to the positions) exactly like C4G system software does;

– "Cartesian Correction Algorithm": software module that communicates with the sensor, receives "actual" Cartesian positions (estimated from motor positions) and "reference" Cartesian positions (estimated from reference Joint position evaluated by C4G system software) from direct kinematics, runs the specific control to realize (impedance, force, hybrid, …) and gives the Cartesian position correction to the software module that manages inverse kinematics.

Considering the above description of software levels running on C4G and on external PC, two important conclusions can be drawn:

a. it's trivial to understand the reason of choosing to implement C4G Open technology in Joint frame instead of in Cartesian frame: C4G Open module acts at the communication software level;

b. about the PC software, except software modules that implement direct and inverse kinematics (which is well known and fixed, according to the class of the chosen robot), the end user only has to deal with the communication with the sensor and with the implementation of the algorithms that implement the required final application.

This modality is not currently available. It is expected for future releases.

14.16 Modes summary

In the following table all C4G OPEN modalities are summarized. The constraints the system must comply with, for each specified modality, are also listed.

Tab. 14.13- Summary of the C4G OPEN Modes

Mode	PC	SMP+	NOTEs
0	--	target and position and velocity loop	default mode
0’	Reserved to COMAU		
1	target and position and velocity loop	--	a., b., c., d., e., f., p.
2	position and velocity loop	position target	b., f., o., p.
3	velocity loop	velocity target	b., c., e., f., g., o.
4	absolute position target	position and velocity loop	a., d., e., f.
5	relative position target	position and velocity loop	a., d., e., f.
6	velocity target	velocity loop	a., c., d., e., f., g.
7	position additional contribute	target and position and velocity loop	c., d., e., f., o.
8	velocity additional contribute	target and position and velocity loop	c., d., f., h., o.
9	current additional contribute	target and position and velocity loop	f., o.
1xx	general measures	target and position and velocity loop	l.
101	accelerometer measures	target and position and velocity loop	f., i., o.
102	videocamera measures	target and position and velocity loop	c., d., e., f., j.
103	force/torque sensor measures	target and position and velocity loop	c., d., e., j., k.
201	cartesian position target	position and velocity loop	d., m.
202	cartesian position delta target	position and velocity loop	d., m., n.

14.16.1 Notes

a. No via PDL2 motion,

b. output saturation current,

c. following error handling,

d. external trajectory control,

e. final position correction,

f. removing the dynamic model by means of C4G restart,

g. position loop deactivation,

h. velocity contribution average value must be zero,

i. current, velocity or delta position contribution,

j. delta position contribution,

k. information handling on SMP+, by means of the impedance model,

l. measure handling on SMP+,

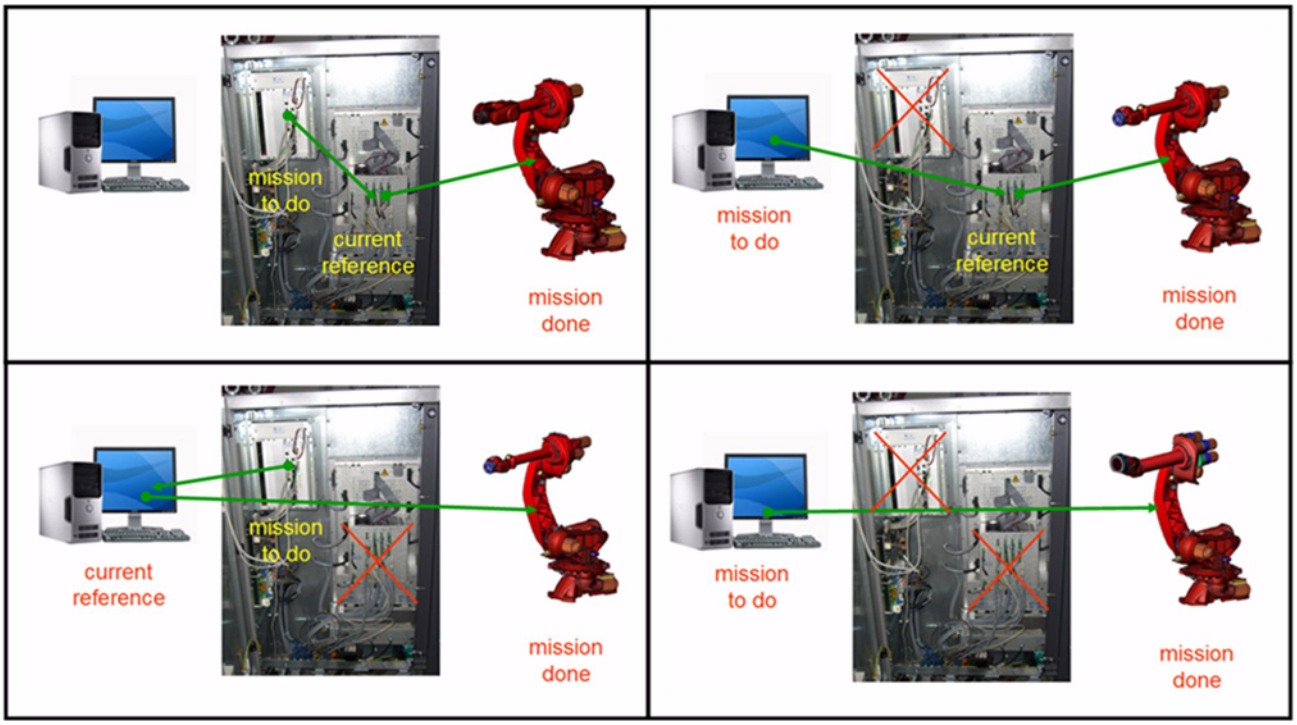
m. kinematic inversion on SMP+,

n. cartesian contribution addition, on SMP+,

o. “simple” control loops,

p. following error control and maximum current value control, from PC.

Fig. 14.8 - Draft representation of the most important OPEN models



leftmost up: Mode 0
rightmost up: Mode 5
leftmost down: Mode 2
rightmost down: Mode 1

14.17 Modes and safety controls

– C4G-side safety controls

– PC-side safety controls.

14.17.1 C4G-side safety controls

In the following table, all C4G internally active safety controls related to each C4G OPEN mode, are listed.

Tab. 14.14- Summary of the available controls for each C4G OPEN mode

Mode	Controls		
	PC-SMP+ communication	current value limit	following error
0	active	active	active
0’	Reserved to COMAU		

Tab. 14.14- Summary of the available controls for each C4G OPEN mode (Continued)

1	active	active	not active **(**)**
2	active	active	active
3	not yet available mode		
4	active	active	active **(*)**
5	active	active	active **(*)**
6	not yet available mode		
7	active	active	active **(***)**
8	active	active	active **(****)**
9	active	active	active **(*****)**
1xx	not yet available mode		
101	not yet available mode		
102	not yet available mode		
103	not yet available mode		
201	not yet available mode		
202	not yet available mode		

For all Open modes the communication control is active: if either the communication between SMP+ and PC drops, or some communication packets are lost or the external PC does not answer back in the predetermined elapsed times, the robot is put into a safety state upon the generation of the corresponding communication error.

To go back to the regular system functioning it is needed to restart C4G.

14.17.2 PC-side safety controls

Please note that the here described safety controls, <u>are the responsibility of the C4G OPEN user</u>.

For any modality in which the external PC either totally or partially generates the trajectory, provided that C4G cannot verify its correctness, it is needed that the suitable stroke-ends control is performed by external PC software. Moreover, it is mandatory that external PC software checks the continuity of the calculated references.

In particular, in the case in which external PC totally or partially supplies the trajectory, without closing the position and velocity loops, the position and velocity references for C4G interpolator are updated at the end of the movement. This means that, when any errors or emergencies occur, the system could exit from the open modality without proper exit conditions, thus also before the movement is accomplished.
It could thus happen that pos_i is not updated to the pos_k currently accumulated value.
In such cases, the robot could show rapid movement transitories caused by the actual difference between pos_i and pos_k current values.
BE EXTREMELY CAREFUL!

The required operations to bring the system to work properly again, are as follows:

a. stop executing any test/user program in the current C4G Open modality;

b. **set system variable $C4G_PA:=1;**

c. **reset the modality change system variables**
($ARM_DATA[idx_arm].C4GOPEN_MODE[idx_axis] := 0
which sets the corresponding axes to the default modality);

d. issue a Configure Save All (CSA command);

e. issue a Cold Restart (CCRC command);

f. **if needed, move the robot to the idle position, BEING EXTREMELY CAREFUL!**

Note that steps b., c. and d. are not needed if the user has not saved (CSA command) any values different from the default ones, for such a system variable.

In any modality in which **PC exclusively supplies the robot trajectory** (marked by one asterisk **(*)**), the following error control is referred to such a trajectory; thus, if it is wrong, C4G cannot detect the error. Anyway, it is still active the maximum suppliable motors current control by current saturation.

In modes in which PC evaluates the coefficients for the position and velocity loops, supplying **SMP+ with the current** (marked by two asterisks **(**)**), the only active control is the one on the maximum suppliable motors current, by means of current saturation.

In the operation modes in which **PC supplies SMP+ with the robot trajectory as an additional component** (marked by three asterisks **(***)**), the following error control is referred to the global trajectory, so, if there is a mistake, C4G cannot detect it. Anyway, the maximum suppliable motors current control by current saturation, and the maximum offset control between a position of the reference trajectory and the previously scheduled one, are still active.

In the operation modes in which **PC supplies SMP+ with the robot velocity as an additional component** (marked by four asterisks **(****)**), the following error control is referred to the trajectory scheduled by SMP+. Anyway, the maximum suppliable motors current control by current saturation, and the maximum offset control between a position of the reference trajectory and the previously scheduled one, are still active.

In the operation modes in which **PC supplies SMP+ with the motors current as an**

additional component (marked by five asterisks **(*****)**), it is still active the maximum suppliable motors current control by current saturation, and the maximum offset between a position of the reference trajectory and the previously scheduled one.

15. SPECIAL MODALITIES

By means of the mechanism to change the C4G OPEN modality, further functionalities can be activated too:

- Active Freezing
- DRIVING ON
- Passive Freezing
- Server unlock
- DRIVE OFF request
- RESTART request
- Following error from PC.

15.1 Active Freezing

When using the *C4GOPEN_MOD_FREEZING_A* (500) special modality, SMP+ no longer checks its communication with PC. In this way it is possible to perform long lasting not real-time operations on PC (such as to read a file) preventing that the system stops because of missing some communication packet.

It is called an **active** freezing, because PC itself freezes the communication and then activates it again.
When SMP+ receives *C4GOPEN_MOD_FREEZING_A* signal, it takes it as a valid packet, even if it does not cause a fatal error if, at the next interpolator tick, PC does not send its packet.
As soon as SMP+ receives a packet from PC, including the EXIT_FROM_OPEN value in its INFPAR field, it activates the communication again, switches the system to Mode 0 and, starting from the next packet, recovers the normal communication with PC.

Attention, please! This particular procedure is to be executed presetting the system to Mode 0, which is the only modality in which the content of the packet, coming from PC, is meaningless for SMP+ actions.

15.2 DRIVING ON

During DRIVE ON procedure it is not allowed (neither by PC nor by PDL2) to set a modality different from 0. To do so, SMP+, while driving ON, forces the special *C4GOPEN_MOD_DRIVING_ON* (501) modality, ignoring any request coming from server.
When the DRIVE ON procedure is accomplished, the modality is no longer *C4GOPEN_MOD_DRIVING_ON*, rather the one set either by PC or PDL2.

Note that server is able to exactely detect the end of the DRIVE ON procedure, simply detecting the moment in which the receiving packet SMax field goes from *C4GOPEN_MOD_DRIVING_ON* to the one set either by PC or PDL2.

15.3 Passive Freezing

Using the special *C4GOPEN_MOD_FREEZING_P_IN* (502) modality, SMP+ no longer sends packets to PC, freezing any activities with PC itself.
In such a way, it is possible to perform long lasting non real-time operations on C4G (e.g.: reading moni.log) preventing the system to stop because of missing communication packets.

It is called a **passive** freezing, because the reactivation of the usual communication, is performed setting the special *C4GOPEN_MOD_FREEZING_P_OUT* (503) modality, by SMP+.
Actually, PC is not aware of that, because it waits for a packet which, during the time interval from the *C4GOPEN_MOD_FREEZING_P_IN* command and the moment the *C4GOPEN_MOD_FREEZING_P_OUT* modality command is not sent by SMP+.

Attention, please! This particular procedure is to be executed presetting the system to Mode 0, **which is the only modality in which the content of the packet, coming from PC, is meaningless for SMP+ actions.**

15.4 Server unlock

As already told, the server process, running on PC, must be activated before activating C4G.
Moreover, server must always be ready to coherently answer to any requests from SMP+.

When it is needed to make PC to take full control again, it is allowed to use the special *C4GOPEN_MOD_EXIT_OPEN* (504) modality. This modality has just been provided to definitively stop the real-time communication between PC and SMP+.
As soon SMP+ receives such a request (either by PDL2 or CCRC command), it sends it to PC. In such a way, at the next interpolator tick, SMP+ no longer checks the real-time modality, whilst the server process, running on PC, can exit the services to client requests infinite loop.

Attention, please! This particular procedure is to be executed presetting the system to Mode 0, which is the only modality in which the content of the packet, coming from PC, is meaningless for SMP+ actions.

15.5 DRIVE OFF request

A special modality is provided, which allows PC to ask SMP+ to execute the DRIVE OFF procedure to stop the robot.
The operating modality is *C4GOPEN_MOD_DRIVE_OFF* (505): as soon as the external PC sends such a value in the communication packet, the system schedules the DRIVE OFF procedure and then switches to Mode 0.

15.6 RESTART request

While implementing a new application running on remote PC, it is useful to be able to often restart the server, preventing to restart C4G every time.
It is then available a special mode which allows just restarting the client service on SMP+, without restarting C4G.

This operating mode is *C4GOPEN_MOD_SVR_RESTART* (506) which can be set by means of PDL2 instructions.

As soon as the communication restarts, obviously the first sent packet is the initialization one. Note that the synchronization packets are not sent again.

15.7 Following error from PC

In Mode 1 and Mode 2 the external PC must check the following error and the motors reference currents values.

In detail, the checks to be performed, are:

$$err(i) = \left| pos_{ref}(i) - pos_{mis}(i) \right| \leq err^{\max}(i)$$

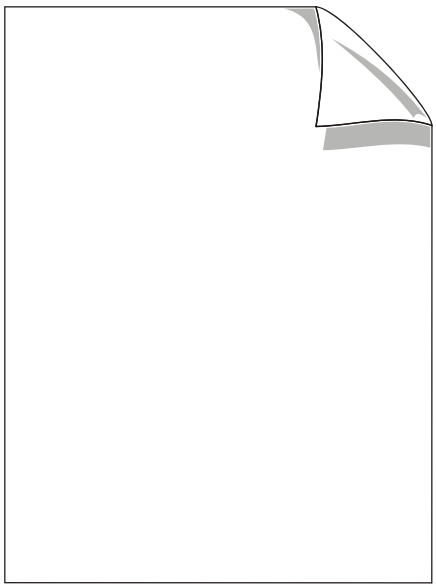
$$\left| cur_{ref}(i) \right| \leq cur^{\max}(i)$$

Where:

- **pos**$_{ref}$**(i)** is the current reference position of **i** axis,
- **pos**$_{mis}$**(i)** is the current measured position of **i** axis,
- **err(i)** is the current following error of **i** axis,
- **err**$_{max}$**(i)** is the following error maximum allowed threshold for **i** axis,
- **cur**$_{ref}$**(i)** is the current reference current of **i** axis,
- **cur**$_{max}$**(i)** is the reference current maximum allowed value for **i** axis.

If the following error exceeds the maximum allowed threshold (at least for one axis), PC asks SMP+ to issue an error, by means of the special *C4GOPEN_MOD_FOLLOW_ERROR* (508) modality.
Obviously, after having issued the error, SMP+ puts the system in DRIVE OFF state.

If the reference current exceeds the maximum allowed value, PC supplies SMP+ with the max allowed value for such a specific axis, performing the remote current saturation.



16. C4G OPEN MONIS

To monitor the C4G OPEN system performance, which means any operations executed by SMP+, any commands issued by PC and any robot answers related to motors reference currents and position values coming from the encoders, several **moni** types are available that keep track of the packets contents, exchanged by SMP+ and PC, at each interpolator tick.

- **The MONI word, stands for the COMAU software tool to analyze any kind of real-time signal, inside C4G.**
- **In the following sections, RX PACKET and TX PACKET terms, stand for, respectively, the received packet from SMP+ and the one transmitted by e SMP+.**

A detailed description follows about:

- Moni Type 19
- Moni Type 20
- Moni Type 21
- Moni Type 22
- Moni Type 23
- Moni Type 24.

16.1 Moni Type 19

For each sampled axis, the following information is provided:

- RX packet - Field d1 [--]
- RX packet - Field d2 [--]
- RX packet - Field d3 [--]
- RX packet - Field d4 [--]
- RX packet - Field d5 [--]
- C4G OPEN modality [--]

This moni type is useful to analyze the whole packet coming from PC.

16.2 Moni Type 20

For each sampled axis, the following information is provided:

- TX packet - Field d1 [--]

– TX packet - Field d2 [--]
– TX packet - Field d3 [--]
– TX packet - Field d4 [--]
– TX packet - Field d5 [--]
– C4G OPEN modality [--]

This moni type is useful to analyze the whole packet coming from SMP+.

16.3 Moni Type 21

For each sampled axis, the following information is provided:

– RX packet - Field d1 [--]
– RX packet - Field d2 [--]
– TX packet - Field d1 [--]
– TX packet - Field d3 [--]
– TX packet - Field d5 [--]
– C4G OPEN modality [--]

This moni type is a mixed one: it keeps track of information included both in the packet coming from SMP+ and coming from PC.
It is useful to analyze Mode 5, because the first two information is, respectively, the position and velocity target, calculated by PC, whereas the last two information is, respectively, the actual position supplied by DSA and the reference current sent to DSA itself. The third information is the target scheduled by SMP+.

16.4 Moni Type 22

For each sampled axis, the following information is provided:

– RX packet - Field d4 [--]
– RX packet - Field d5 [--]
– TX packet - Field d1 [--]
– TX packet - Field d3 [--]
– TX packet - Field d5 [--]
– C4G OPEN modality [--]

This moni type is a mixed one: it keeps track of information included both in the packet coming from SMP+ and coming from PC.
It is useful to analyze Mode 2, because the third information is the target calculated by SMP+, the fourth information is the actual measured position coming from the encoders; the first and the second information is the velocity and current contributions, calculated by PC, whereas the last information is the motors reference current.

16.5 Moni Type 23

For each sampled axis, the following information is provided:

– TX packet - Field d1 [--]
– TX packet - Field d5 [--]
– TX packet - Field e1 [--]
– TX packet - Field e2 [--]
– RX packet - Field d5 [--]
– C4G OPEN modality [--]

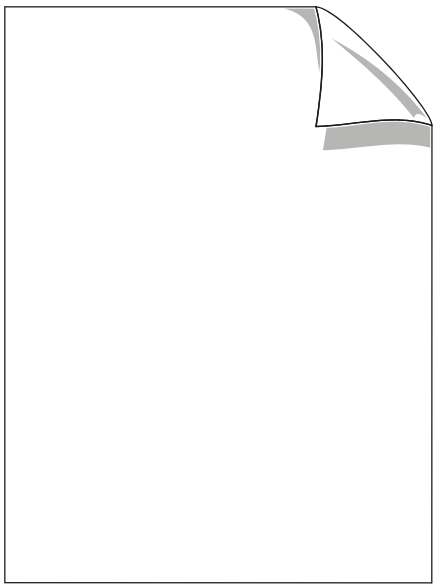
This moni type is a mixed one: it keeps track of information included both in the packet coming from SMP+ and coming from PC, furthermore it keeps track of data included in the EXT fields.
It is useful to analyze Mode 9, because the first four information (position target, reference current, dynamic model current and diagonal inertia) are needed for PC to calculate the additional current to be supplied to DSA. This datus is kept in the fifth information of this moni type.

16.6 Moni Type 24

For each sampled axis, the following information is provided:

– TX packet - Field d1 [--]
– TX packet - Field d5 [--]
– TX packet - Field e1 [--]
– TX packet - Field e2 [--]
– RX packet - Field d4 [--]
– C4G OPEN modality [--]

This moni type is a mixed one: it keeps track of information included both in the packet coming from SMP+ and coming from PC, furthermore it keeps track of data included in the EXT fields.
It is useful to analyze Mode 8, because the first four information (position target, reference current, dynamic model current and diagonal inertia) are needed for PC to calculate the additional velocity to be supplied to DSA. This datus is kept in the fifth information of this moni type.



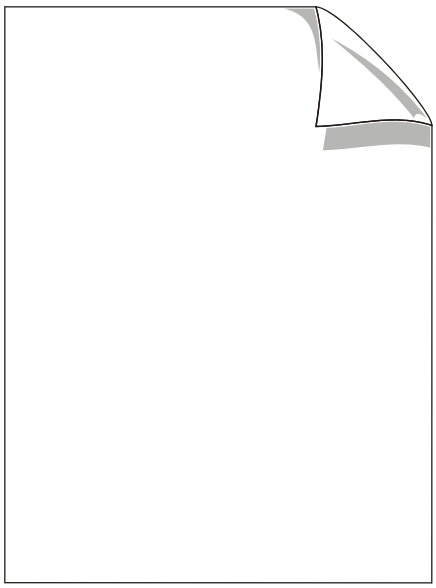
17. C4G OPEN ERRORS

An error messages list is shown below, related to the errors generated by SMP+ in case of C4G Open malfunctioning conditions.

In the case in which any different kinds of error occur, please refer to the Standard C4G System manuals.

Tab. 17.1 - C4G OPEN errors

Error	Message	Description
ERR_C4GOPEN_SKT_UDP	*C4GOpen: error opening the socket UDP port 1000*	It is not possible to set up a communication with the server, because it is impossible to open a UDP socket on 1000 port
ERR_C4GOPEN_UDP	*C4GOpen: communication error*	The C4G did not receive the UDP packet from PC for 10 consecutive times
ERR_C4GOPEN	*C4GOpen: configuration error*	Too many axes are being configured in an open modality
ERR_C4GOPEN_CONFIGRATION_AX	*C4GOpen: Arm %d Axis %d not present*	The user is trying to configure the Open modality on an axis not configured for this
ERR_C4GOPEN_UDP_DEBUG	*C4GOpen: Rx Debug*	C4G did not receive the packet coming from PC
ERR_C4GOPEN_TX_UDP	*C4GOpen: error in socket UDP transmission*	It is not possible to send the UDP packet
ERR_C4GOPEN_FOLLOW_ERROR	*C4GOpen: arm %s axis %d Following error out of range*	PC cannot maintain the following error under a certain limit. This is due to the value of the currents coming from PC
ERR_C4GOPEN_OPT_HW	*C4GOpen: software option not allowed for arm %d*	It is not possible to enable the C4GOpen option on this controller
ERR_C4GOPEN_DRIVEOFF_SERVER	*C4GOpen: Drive Off from Pc*	DRIVE OFF command from pc
ERR_C4GOPEN_SET_0_MODALITY	*C4GOpen: arm %d , forced 0 modality*	The 0 modality has been forced after switching the state selector from AUTO to PROG or from PROG to AUTO when the modality was different from 0.



18. C4GOPEN_CMD VARIABLE

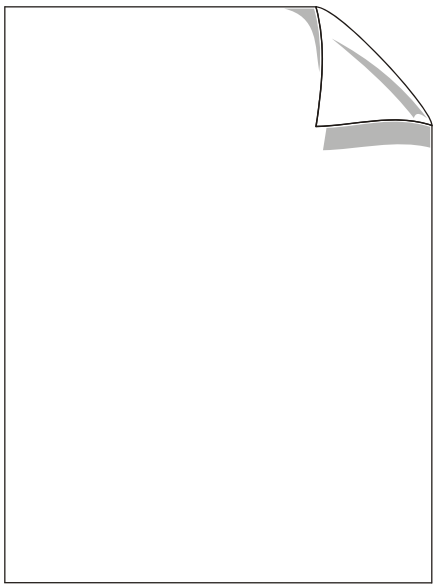
The system variable $ARM_DATA[$i_{ARM}$].C4GOPEN_CMD is a 16 elements array to be used for some service functions.

In the current section, it is summarized the meaning of the currently used elements:

– *$ARM_DATA[$i_{ARM}$].C4GOPEN_CMD[1]*: it specifies the motion type within a complex trajectory to be executed just in one modality;

– *$ARM_DATA[$i_{ARM}$].C4GOPEN_CMD[2]*: it specifies the modality change.

ATTENTION, please! This variable is not automatically reset by the system, thus, if it has to be monitored, the user should better set its value to 0 as soon as it goes to 1.



19. TRANFORMATIONS UNDER KINEMATIC INFLUENCES

Fig. 19.1 shows the diagram of the kinematic influences which could be present in the wrists of some COMAU robots.

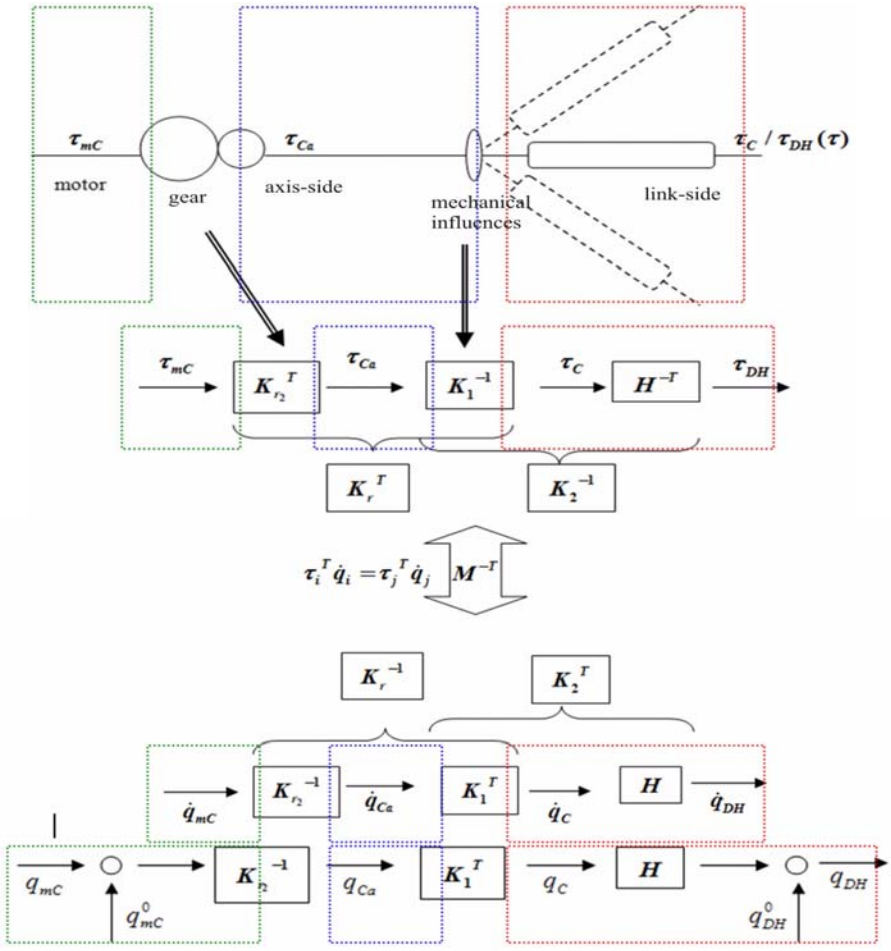
To analitically express the existing relationships between the calculated data in several positions of the kinematic chain, it is useful to know the following definitions:

– **motor-side variables** ("mC"): these quantities are calculated motor-side, in general expressed according to COMAU standards;

– **drive variables** ("Ca"): these quantities are calculated axis-side, in general expressed according to COMAU standards;

– **link-side variables k** ("C" o "DH"): these are link-side quantities, expressed, depending on the specific case, according to either COMAU ("C") or Denavit-Hartenberg ("DH") standards.

Fig. 19.1 - Wrist kinematic influences Diagram



Moreover, it is to be outlined that if the quantities are positions, the encoder data (obviously expressed motor-side) must be depurated of the $\mathbf{q}^0_{mc}$ calibration constants,

calculated in motor turns.

Finally, as far as the link positions, the relationships between COMAU standards and Denavit-Hartenberg standards, must take into account the existing offsets between the two reference systems, as well as the transformation matrix H.

K_{r_2} matrix which associates motor-side data to drives data, is called a **pure transmission ratios matrix** (where “pure” means without the kinematic influence coefficients) and, for a six axes robot, looks as follows:

$$K_{r_2} = \begin{bmatrix} k_{r_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & k_{r_2} & 0 & 0 & 0 & 0 \\ 0 & 0 & k_{r_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & k_{r_4} & 0 & 0 \\ 0 & 0 & 0 & 0 & k_{r_5} & 0 \\ 0 & 0 & 0 & 0 & 0 & k_{r_6} \end{bmatrix}$$

where k_{r_i} elements are the pure transmission ratios (with sign) of reducers or gears used by each of the 6 axes.

$\mathbf{K}_1$ matrix, which associates drives data to link-side data, is called **kinematic influences matrix** and, for a 6 axes robot, looks as follows:

$$K_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & k_{54} & k_{64} \\ 0 & 0 & 0 & 0 & 1 & k_{65} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where:

– k_{54} is the axis 4 influence coefficient on axis 5 (i.e. moving axis 4, axis 5 moves too, according to the k_{54} factor),

– k_{64} is the axis 4 influence coefficient on axis 6 (i.e. moving axis 4, axis 6 moves too, according to the k_{64} factor),

– k_{65} is the axis 5 influence coefficient on axis 6 (i.e. moving axis 5, axis 6 moves too, according to the k_{65} factor).

Their values are calculated according to the internal gear in the kinematic chain which can be determined by means of the following fields:

```
arm1AxInfo[GBM_NAX_ARM]
arm2AxInfo[GBM_NAX_ARM]
```

```
arm3AxInfo[GBM_NAX_ARM]
arm4AxInfo[GBM_NAX_ARM]
```

and of the initialization packet.

To better explain:

```
k54 = arm*AxInfo[4]
k64 = arm*AxInfo[5]
k65 = arm*AxInfo[6]
```

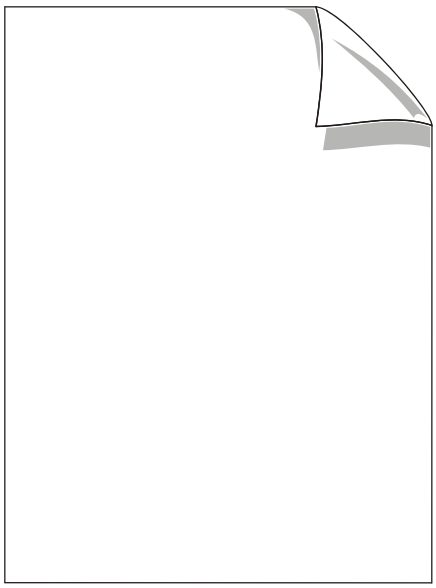
$\mathbf{K}_r$ matrix, which directely associate the motor-side data to the link-side data, is called **transmission matrix** and is calculated as follows:

$$K_r = K_{r_2} K_1^{-T} :$$

$$K_r = \begin{bmatrix} k_{r_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & k_{r_2} & 0 & 0 & 0 & 0 \\ 0 & 0 & k_{r_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & k_{r_4} & 0 & 0 \\ 0 & 0 & 0 & -k_{r_5} k_{54} & k_{r_5} & 0 \\ 0 & 0 & 0 & k_{r_6}(k_{65}k_{54} - k_{64}) & -k_{r_6} k_{65} & k_{r_6} \end{bmatrix}$$

For example, to transform link-side positions in radians, to motor-side positions in motor turns (including the calibration constants), the following formula is to be used:

$$q_{m_C} = K_r \frac{q_C}{2\pi} + q_{m_C}^0$$



20. SYNCHRONISM SIGNAL

In order to accurately synchronize PC and C4G time basis, it is possible to provide PC with the network synchronization signal (SYNC), produced by SMP+ and used by DSA.

This hardware synchronization mechanism is added to the software one, based on the SMP+ and PC packets real-time communication.

The SYNC signal is a 8KHz (125us) square wave.

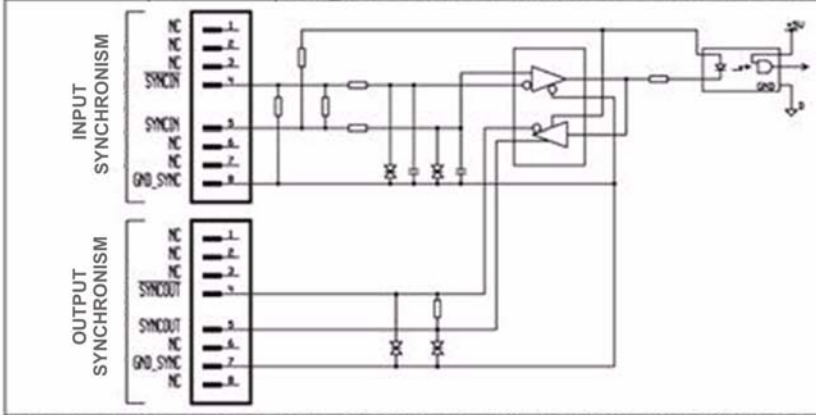
From the Physical point of view, such a signal is transmitted by means of a 3 wires connection: one wire as 0V reference for signals and 2 differential wires for the signal.

The SYNC IN (X6) connector (see Fig. 20.1) is used for the external clock input and takes it internally to the drive, the SYNC OUT (X5) connector is the same clock signal output, so that the synchronization signal is available to any other drives or, in case of C4G OPEN, to the external PC.

Obviously, if some more devices need the hardware synchronization signal, they must pass the SYNC signal one to the other, since they are enabled to use the DAISY CHAIN configuration.

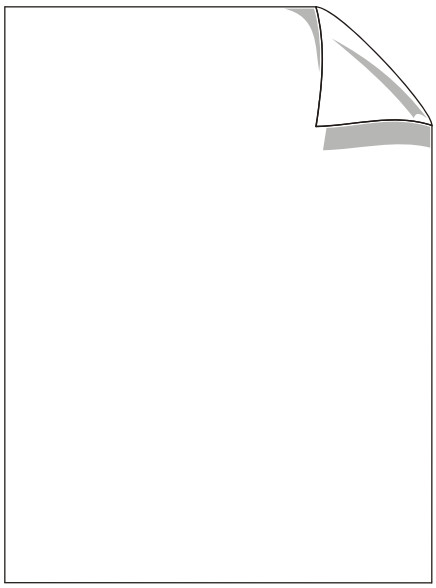
Fig. 20.1 - SYNC signal electrical diagram.

	Position	Function
SYNC OUT	1, 2, 3, 6, 8	Not Connected
	4	SYNCOUT-: signal received from the drive, by means of an RS485 standard serial port (in differential configuration)
	5	SYNCOUT+: see above
	7	GND_SYNC: input of the synchronism ground signal
SYNC IN	1, 2, 3, 6, 7	Not Connected
	4	SYNCIN-: signal sent to the drive, by means of an RS485 standard serial port (2 wires in differential configuration)
	5	SYNCIN+: see above
	8	GND_SYNC: input of the synchronism ground signal



Two RJ45 standard fixed connectors - 8 TYCO ELECTRONICS 406525-1 (cod. MOOG AK5801) pins

RJ45 standard plug - 8 TYCO ELECTRONICS 6-554169 (cod. MOOG AK4750) pins



21. TIMING

Before talking about the timing C4G OPEN functioning is based on, it is needed to give some definitions:

- PC basic period $\mathbf{T_{PC}}$: it is the PC basic period for statements execution;
- basic synchronization period $\mathbf{T_{SYNC}}$: it is the period of the synchronization signal between SMP+ and DSA, 125usec (8KHz) (see Chap.20. - Synchronism signal);
- SMP+ basic period $\mathbf{T_{SMP+}}$: it is the interpolator sampling period, 1msec (1KHz) in case of C4G Open modality;
- DSA basic period $\mathbf{T_{DSA}}$: it is the position and velocity loops sampling period, inside DSA, 500usec (2KHz) in case of C4G Open modality;
- DSP basic period $\mathbf{T_{DSP}}$: it is the current loop sampling period, inside DSP, 62.5usec (16KHz);
- $\mathbf{T_{CYCLE2}}$ interval: it is the maximum time interval (less than 2.5ms) between the reception of the encoders position data and the moment in which DSA actually executes the action calculated by PC;
- $\mathbf{T_{CYCLE1}}$ interval: it is the maximum time interval for the PC to be able to answer back to the packet sent by SMP+, 500-600usec;
- robot delay $\mathbf{T_{ROBOT}}$: it is the delay for the robot mechanics to respond to the positional references.

Fig. 21.1 - C4G OPEN Timing

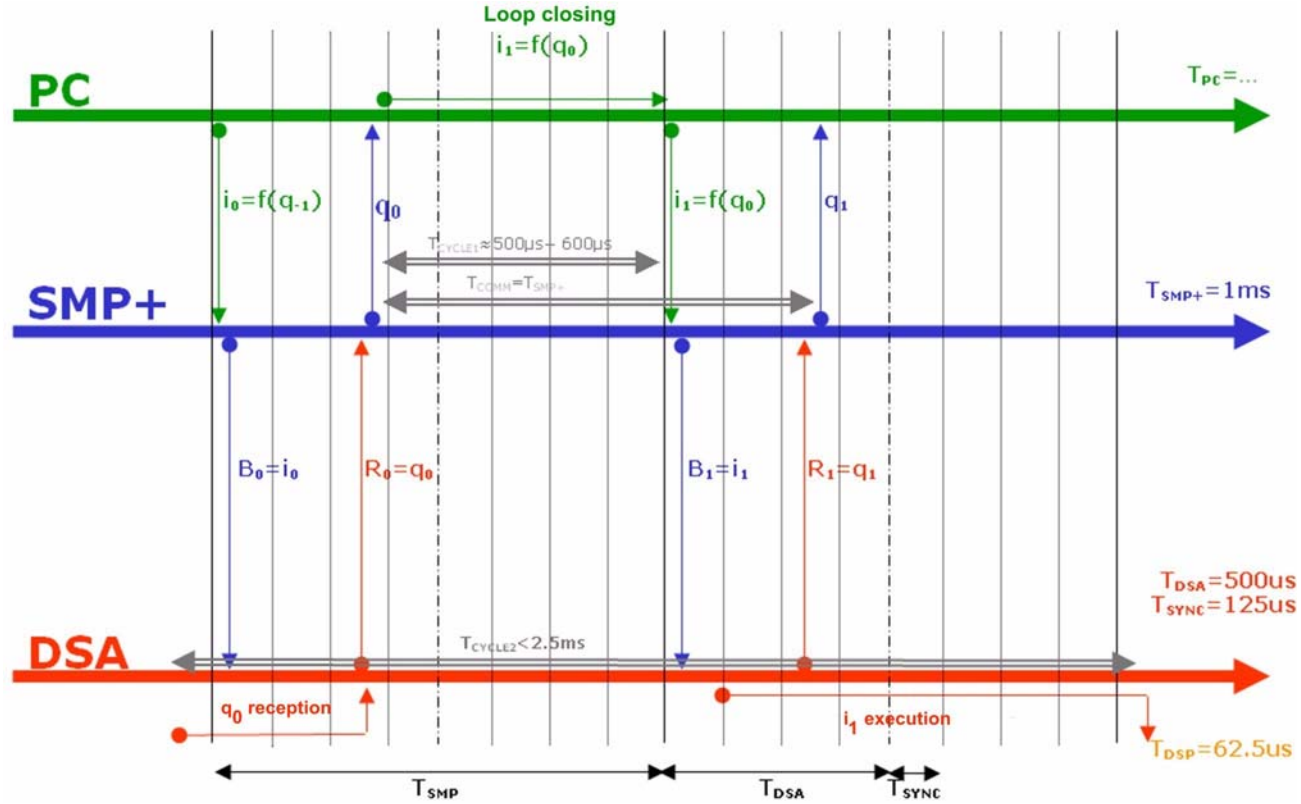


Fig. 21.1 shows the timing diagram which is the time basis for the SMP+, DSA and PC communication.

The case has been considered, in which PC must process the positions evaluated by the encoders and, depending on them, calculate the general contributions (such as reference currents, references positions, velocity references, dynamic model contrinution, etc.) to be sent to DSA.

When the motor position has been measured by the encoder, DSA does not immediately send such an information to SMP+, but just after the second sequential synchronization tick ($\mathbf{T}_{SYNC}$), moreover, before the half next SMP+ tick ($\mathbf{T}_{SMP+}$) in DSA-->SMP+ (RAAZ) packet.

As soon SMP+ receives the information, it sends it to PC. At the beginning of the next interpolation tick, SMP+ ($\mathbf{T}_{SMP+}$) must provide DSA with any information in the SMP+-->DSA (BLIP) packet.
Obvioulsly, if the PC answer is on time, SMP+ is able to supply DSA with any information coming from PC itself.
This is the reason for PC to answer back to SMP+ in a time which is less than $\mathbf{T}_{CYCLE1}$.

When DSA receives BLIP packet from SMP+, the included information is not immediately processed by DSP, but just at the beginning of the second sequential DSA tick ($\mathbf{T}_{DSA}$), moreover, at the beginning of the next SMP+ tick ($\mathbf{T}_{SMP+}$).

Please, note that the time between the motor position measuring and the execution of a PC processed action, is $\mathbf{T}_{CYCLE2}$.

Summarizing, C4G OPEN funtioning is based upon the following infinite cycle actions:

a. PC receives from SMP+ the packet including the latest motor position values (this message also acts as interrupt and PC software synchronization);

b. PC must process such information, before sending it to SMP+, within a $\mathbf{T}_{CYCLE1}$ time interval (if the timeout expires, the cycle is interrupted and a fatal error is issued);

c. SMP+ receives the packet from PC, processes it (depending on the specific open modality) and sends it to DSA;

d. DSA receives the packet (BLIP) from SMP+ and sends back the performed measures;

e. SMP+ processes the received packet (RAAZ) and sends it to PC;

f. the cycle continues to step a. again.

It is then obvious, that the whole cycle execution time is $T_{COMM}=T_{SMP+}=1ms$.

This means that, to optimize the PC actions handling, we should better separate their process into **two phases** (see Fig. 21.2):

– **“algebraic calculation” phase**: during this phase all calculations, based upon the latest motor position coming from DSP, are performed. This phase cannot last more than $\mathbf{T}_{CYCLE1}$.
– **“dynamic calculation” phase**: during this phase, any quantity can be updated, which is based upon the system state, independently from the latest position coming from DSPs. This phase can last at least $\mathbf{T}_{COMM}$-$\mathbf{T}_{CYCLE1}$.

Fig. 21.2 - Pre-calculations and calculations phase on PC

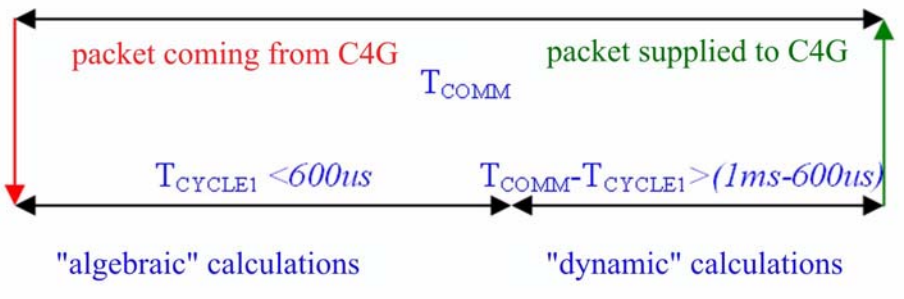
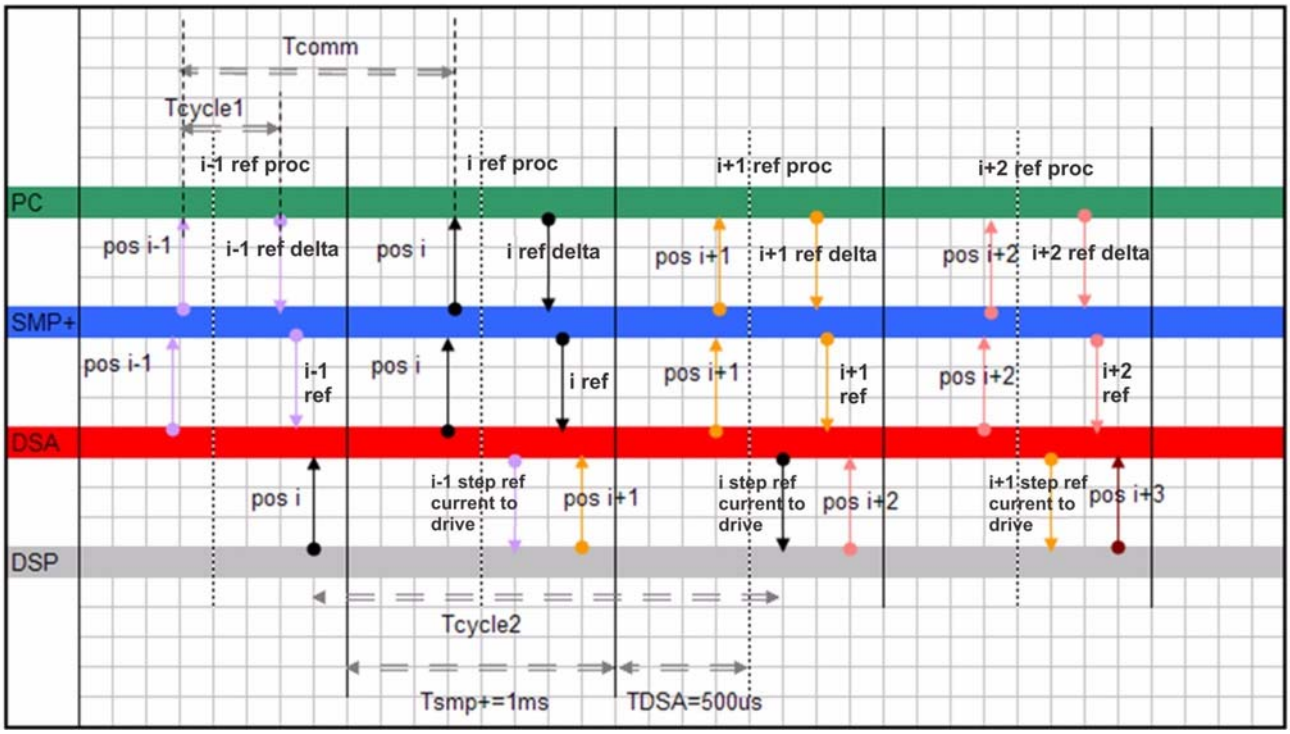
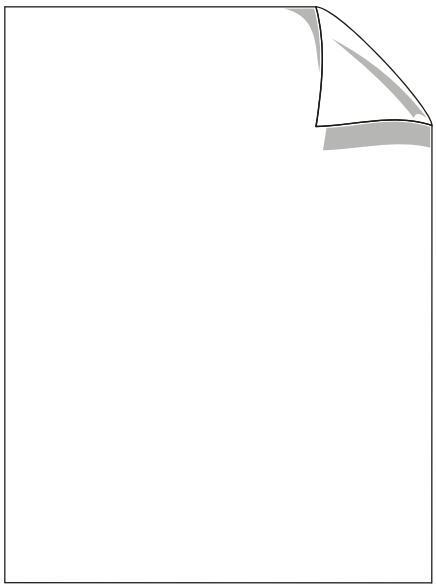


Fig. 21.3 - C4G OPEN model - Information Time scheduling





22. PC CALCULATED TIMING

In the current Chapter a program section is reported which could be executed by PC, to verify the communication timings with SMP+.

```
//***************************************************************************
// AT BEGINNING
//***************************************************************************

// definition of packet period
  double interPacketsPeriod = 1e-3; // C4G Open <-> PC Inter-Packets Period [secs]


//***************************************************************************
// IN CYCLE
//***************************************************************************

// receiving
  si_sts = mvfi_ReceiveFromSocketUDP(&(sx_conn));
  t1 = rt_get_time_ns();

// calculation

// sending
  si_sts = mvfi_SendFromSocketUDP(&(sx_conn));
  t2 = rt_get_time_ns();

// cycle time
  lastCycleTime = (double)(t2 - t1) / 1000000000.0;

// inter-step time
  lastInterStepTime = ((double)(t1 - t1Old)) / 1000000000.0;

// updating statistics
  worstCycleTime = max(lastCycleTime);
  minInterStepTime = min(lastInterStepTime);
  maxInterStepTime = max(lastInterStepTime);

//***************************************************************************
// AT END
//***************************************************************************

// final statistics
  double minJitter = minInterStepTime - interPacketsPeriod;
  double maxJitter = maxInterStepTime - interPacketsPeriod;
  double minPercJitter = minJitter / interPacketsPeriod * 100.0;
  double maxPercJitter = maxJitter / interPacketsPeriod * 100.0;
// display statistics
  printf("Nominal Inter-Packets Period = %.9f s\n", interPacketsPeriod);
  printf("Worst cycle time = %.9f s\n", worstCycleTime);
```

```
printf("Measured Inter-Packets Period range = [%+.9f; %+.9f] s\n",
minInterStepTime, maxInterStepTime);
printf("Jitter range = [%+.9f; %+.9f] s\n", minJitter, maxJitter);
printf("Percentage jitter range             = [%+.9f; %+.9f] %\n\n",
minPercJitter, maxPercJitter);
```

Fig. 22.1 - Timing from the PC point of view

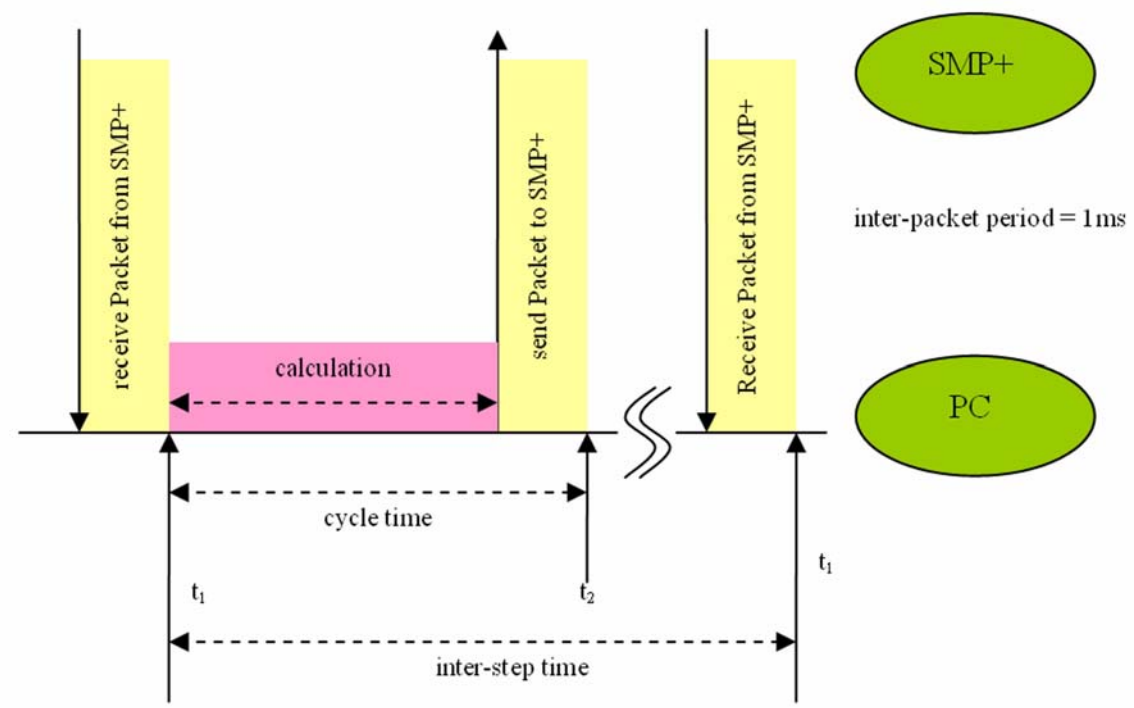


Fig. 22.1 shows the detailed timing of PC operations. Please, note that:

$$\begin{cases} last\_cycle\_time = t_2(k) - t_1(k) \\ worst\_cycle\_time = \max(last\_cycle\_time) \end{cases}$$

$$\begin{cases} last\_inter\_step\_time = t_1(k) - t_1(k-1) \\ min\_inter\_step\_time = \min(last\_inter\_step\_time) \\ max\_inter\_step\_time = \max(last\_inter\_step\_time) \end{cases}$$

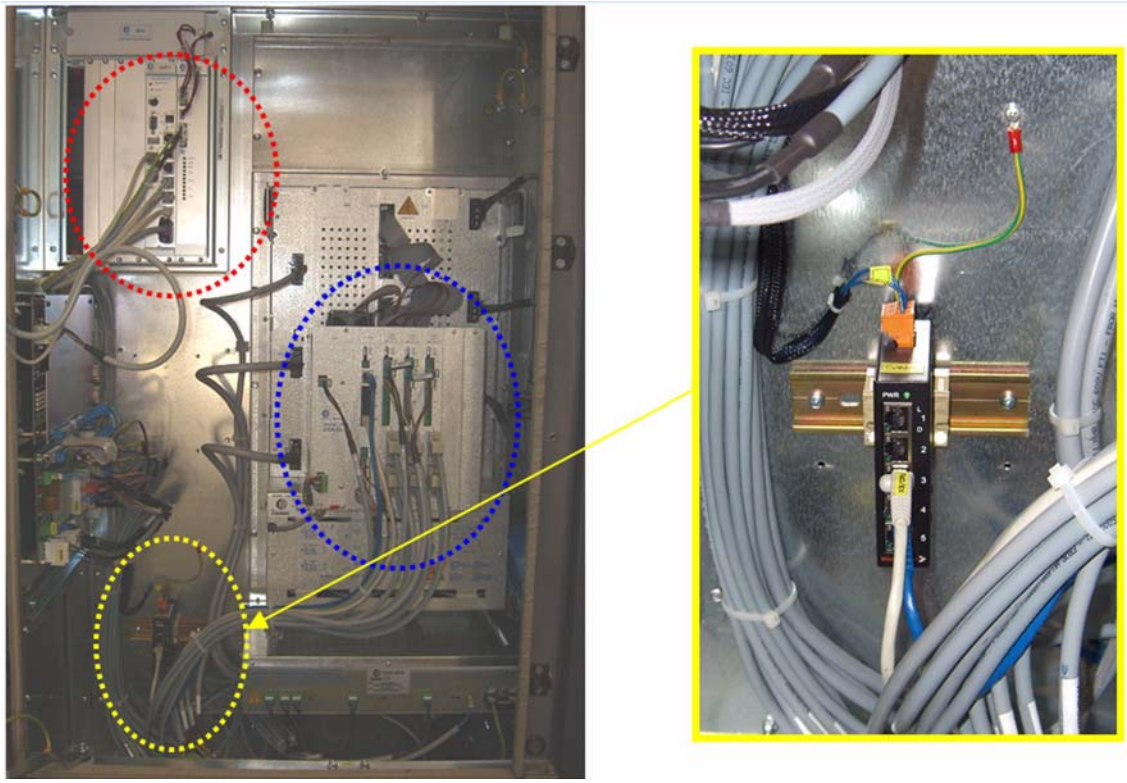
last_cycle_time variable represents the PC time to answer back to SMP+, after receiving the request packet from it.
last_inter_step_time is the time interval between a request reception and the next one. The values of *last_inter_step_time* approache *inter_packet_period* (which corresponds to SMP+ sample time), with the exception of a minimal jitter (operations are led by a hardware interrupt).

23. SWITCH

Fig. 23.1 shows the hardware implementation of the functional architecture included in the blocks of Fig. 5.1 - Open Controller.

The standard switch used by COMAU to implement C4G OPEN, is shown in the right side of Fig. 23.1.

Fig. 23.1 - C4G Open Switch

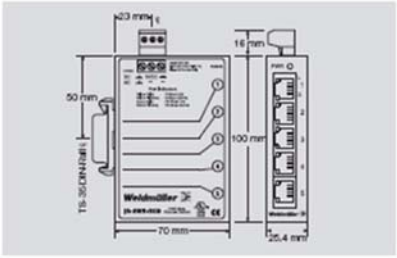


The Switch circuit diagram is provided in the following Fig. 23.2.

Fig. 23.2 - Switch WEIDMULLER mod. IE-SW5-ECO (*)

IE-SW5-ECO

5 Ports



Technical data

Housing		Aluminium
Ports		5xRJ45
AC input voltage, min.-max.		8-24 V AC
DC input voltage, min.-max.		10-36 V DC
AC input power/DC input power		5 VA AC/5 Watt DC
Input frequency		47 - 63 Hz
Operating temperature, min.-max.		0 °C ... 60 °C
Storage temperature, min.-max.		-40 °C ... 85 °C
Installation		DIN Rail Mount TS 35
Ingress Protection Class		IP 20
Standard		ANSI / IEEE 802.3
Data rate		10BASE-T or 100BASE-TX
Segment length		Copper Cable 100m max.
Functionality		Autonegotiation, Autocrossing
Flow control		semi-duplex/full duplex
Status indication		Power; Connection/Activity
Approvals		UL508/CE/EMC/EN 55024 und EN 55022
Aging		300 s
Length x width x height	mm	70 x 25.5 x 100

(*) - extracted from **5652480000_Ethernet-Unmanaged_Switches.pdf document** (free download from WEIDMULLER site).

24. STATE MACHINE

The current chapter fully describes the C4G Open system **Finite State Machine**.

It is divided into the following paragraphs:

– General issues

– Implementation

– Operating description.

24.1 General issues

Fig. 24.1 shows the Finite State Machine (**FSM**) which leads the C4G OPEN functioning, pointing up how some transitions are handled:

– standard open modality (PC control),

– closed modality (SMP+ standard control),

– mixed modality (both PC and SMP+ control).

Listed below are the states definitions:

– S_{00}) System in DRIVE ON state, the robot is not moving (**STOP**) (all interpolators are stopped at the default node)

– S_{10}) the robot is moving under SMP+ control (**CLOSED**)

– S_{01}) the robot is moving under PC control (**OPEN TYPE A**)

– S_{11}) the robot is moving under both SMP+ and PC control (**OPEN TYPE B**)

The transitions are as follows:

– A_{0010}) *MOVE* via PDL2

– A_{1000}) **DELAY**/**WAIT** FOR SIGNAL, …

– A_{0001}) **GO OPEN TYPE A** (open modality with no trajectories supplied by SMP+)

– A_{0100}) **EXIT FROM OPEN** (signal coming from PC)

– A_{0011}) **GO OPEN TYPE B** (open modality with trajectory supplied SMP+ too)

– A_{1100}) NA (two distinct events are needed: SMP+ trajectory end, PC activities end)

– A_{1011}) NA (mixed modality which can only be activated when the robot is not moving)

– A_{1110}) **EXIT FROM OPEN** (signal coming from PC)

– A_{1101}) **END MOVE** (end of SMP+ supplied motion)

– A_{0111}) NA (mixed modality which can only be activated when the robot is not moving)

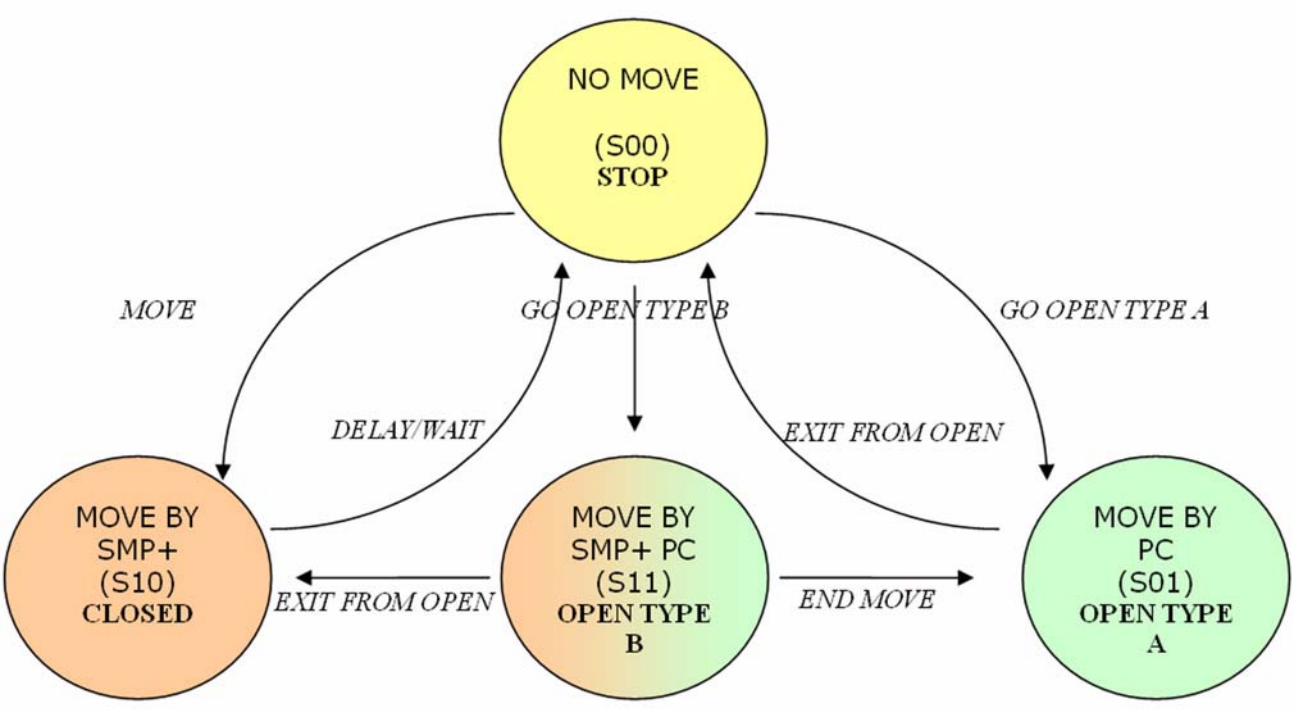
– A_{1001}) NA (open modality which can only be activated when the robot is not moving)

– A_{0110}) NA (exiting from open modality, just allowed when the robot is not moving).

Please note that some state transitions are not allowed. This is due to some C4G OPEN

functioning limitations which require to always pass through the steady state (DRIVE ON and not moving machine) before changing the machine operating modality, as fully described in Chap.13. - Dynamic handling of the OPEN modality.

Fig. 24.1 - C4G OPEN FSM



24.2 Implementation

The shown above diagram, is the conceptual idea which is the C4G Open functioning basis.

In current paragraph, the FSM actual implementation is described.

First of all, the involved “software entities” in the C4G Open implementation, are to be defined:

– **Server PC** (**SW_PC**): code of the server which is running on the external PC and implements all external PC actions,

– **PDL2 Program** (**SW_PDL**): software running on C4G, implementing any PDL2 dedicated instruction to handle the C4G Open functionality,

– **Task SMP+** (**SW_SMP**): task running on SMP+, implementing C4G Open in the System Software.

The listed above software entities, execute all or just some of the following operations:

– synchronization between the different software modules,

– trajectory generation,

– axis control,

– operating mode changing.

More in detail, in the current paragraph the word synchronization stands for the whole signals which allow to synchronize one another the different software entities:

– **SW_PC**:
  - *sent signals*: EXIT_FROM_OPEN signal, to exit from a modality;
  - *received signals*: various open modalities, set by other software entities;

– **SW_PDL2**:
  - *sent signals*: routine to set the different open modalities;
  - *received signals*: "SIGNAL" received on a "WAIT" associated to a semaphore set to Mode 0 (automatically set modality, except some situations, from SW_SMP when receiving EXIT_FROM_OPEN signal from SW_PC);

– **SW_SMP**:
  - *sent signals*: Mode 0 (or some others) setting, upon an EXIT_FROM_OPEN signal, received from SW_PC;
  - *received signals*: receiving EXIT_FROM_OPEN signal from SW_PC and different open modalities setting, from SW_PDL.

As far as the trajectory generation, it is important to outline that the software entity, at a certain moment generating the trajectory, also handles the synchronization for the other software entities.

Moreover:

– **SW_PC**: is able to generate the trajectory either autonomously (Mode 1, Mode 4, Mode 5) or together with SW_PDL2 (e.g. Mode 7);

– **SW_PDL2**: is able, by means of the MOVE statement, to generate the trajectory either autonomously or together with SW_PC (e.g. Mode 7);

– **SW_SMP**: never generates the trajectory, so it is not able to synchronize the other entities, without receiving a synchronization signal before, from one of them.

As far as the axis control, it is always performed by the DSA software, unless when either Mode 1 or Mode 2 is entered.
Mode 1 allows PC to generate the trajectory too, which means that the synchronization full control is up to SW_PC.
Viceversa, in Mode 2, the axis control is up to SW_PC, whereas the trajectory generation is not. In such a case, then, the operations synchronization is up to SW_PDL.

Finally, as far as the operating mode changing, it usually involves three distinct phases:

– initial transitory to enter the modality (**START**),

– running in the modality (**CRUISE**),

– final transitory to exit from the modality (**END**).

A detailed description follows about the synchronization signals to start and/or end such phases for each operating modality.

– Mode 0

– Mode 0'

– Mode 1

– Mode 2

– Mode 4

– Mode 5

– Mode 7

– Mode 8

– Mode 9.

24.2.1 Mode 0

- START: begins either when SW_PDL sets this mode or when exiting from other operating mode, by means of techniques that are described later on.
- CRUISE: not present, because the “running” execution starts as soon as entering this mode.
- END: exiting from the current mode, occurs when SW_PDL sets a different mode.

24.2.2 Mode 0'

- START: begins with SW_PDL setting this mode.
- CRUISE: not present, because the “running” execution starts as soon as entering this mode.
- END: exiting from the current mode, occurs when SW_PDL sets a different mode.

24.2.3 Mode 1

- START: begins when SW_PDL sets this mode. In this phase the axis control switches from DSA to external PC. In the SW_SMP there is a counter to drive this switching phase: it is incremented up to a value which is enough to stabilize the control switch. The current supplied by SW_PC must be able to keep the robot hold during a time interval correlated to the SW_SMP counter, so that the control switching is made while the machine is not moving.
  In fact, in such a modality, SW_PC generates the trajectory, so it takes the control of the situation.
- CRUISE: begins when the SW_SMP internal counter reaches the maximum value and when SW_PC starts generating a trajectory to move the robot.
- END: the final transitory (to exit from current mode), is started by SW_PC sending the EXIT_FROM_OPEN message; SW_PC is the sole keeper of the synchronization, since it generates the trajectory.
  This message sending, causes the system to switch to ‘1 STAND_BY’ auxiliary modality, during which SW_PC must supply the suitable current to keep the robot hold, while SW_SMP starts to swhitch the control from external PC to DSA, decreasing the previously incremented counter. As soon as the counter goes to zero, the modality changes to Mode 0.

24.2.4 Mode 2

- START: begins when SW_PDL sets such a mode. During this phase, the axis control switches from DSA to external PC. A SW_SMP counter exists to drive this switching phase: it is incremented up to a suitable value to make this control passage stable. SW_PDL must include a DELAY statement, set to a value related to the SW_SMP counter, so that the control switching is performed with not moving robot. In fact, while in this mode, SW_PDL generates the trajectory, so it leads the synchronization.
- CRUISE: begins when both the SW_PSMP internal counter reaches its maximum value and the DELAY statements (set by SW_PD) finish; so SW_PDL starts to generate the trajectory which moves the robot.

– END: the final transitory (to exit from current mode), is started by SW_PC setting the ‘2 STAND_BY’ auxiliary mode. SW_PC is the sole keeper of the synchronization, since it generates the trajectory. During this modality, SW_PDL stops on a DELAY statement which is set like the previous one, allowing to go back to DSA axis full control. SW_SMP switches the control by decreasing the previously incremented counter. As soon as such a value goes to zero, the system exits from this mode and switches back to Mode 0.

24.2.5 Mode 4

– START: begins when SW_PDL sets this mode.

– CRUISE: not present, because the “running” execution starts as soon as entering this mode.

– END: exiting from the current mode, occurs when SW_PC sends an EXIT_FROM_OPEN signal.

24.2.6 Mode 5

– START: begins when SW_PDL sets this mode.

– CRUISE: not present, because the “running” execution starts as soon as entering this mode.

– END: exiting from the current mode, occurs when SW_PC sends an EXIT_FROM_OPEN signal.

24.2.7 Mode 7

– START: begins when SW_PDL sets this mode.

– CRUISE: not present, because the “running” execution starts as soon as entering this mode.

– END: exiting from the current mode, starts when an EXIT_FROM_OPEN signal is sent by SW_PC (because SW_PC could be generating the trajectory together with SW_PDL); two different behaviours are possible:
  • SW_PDL has accomplished the scheduled movement: the system immediately enters ‘7 STAND_BY_EXIT’ auxiliary mode and then Mode 0,
  • SW_PDL has not accomplished the scheduled movement: the system passes through ‘7 STAND_BY’ auxiliary mode. In such a mode SW_PC no longer supplies the trajectory, so the trajectory full control, thus the synchronization full control, is given to SW_PDL. As soon as the movement finishes, the system enters ‘7 STAND_BY_EXIT’ auxiliary mode and finally Mode 0.

24.2.8 Mode 8

– START: begins when SW_PDL sets this mode.

– CRUISE: not present, because the “running” execution starts as soon as entering this mode.

– END: exiting from the current mode, starts when an EXIT_FROM_OPEN signal is sent by SW_PC.

24.2.9 Mode 9

– START: begins when SW_PDL sets this mode.

– CRUISE: not present, because the “running” execution starts as soon as entering this mode.

– END: exiting from the current mode, starts when an EXIT_FROM_OPEN signal is sent by SW_PC.

24.3 Operating description

The meanings of synchronization, trajectory generation, axis control and mode switching related to the three software entities involved in the C4G OPEN option (server running on external PC, PDL2 motion program and task running on SMP+), are schematically outlined in the following Fig. 24.2, Fig. 24.3 and Fig. 24.4).

The first one shows the states machine driving SW_SMP, the second one SW_PC actions and the third one SW_PDL commands.

As far as the chosen colours in the following figures,

– **blue indicates SW_SMP machine states and the occurring events coming from the counters this software uses for synchronizations,**

– **red indicates SW_PDL settings, related to the different modes,**

– **black are SW_PC actions, related to the currently set mode in a particular moment,**

– **green indicates EXIT_FROM_OPEN signals generated by SW_PC.**

Fig. 24.2 - State machine SW_SMP-side

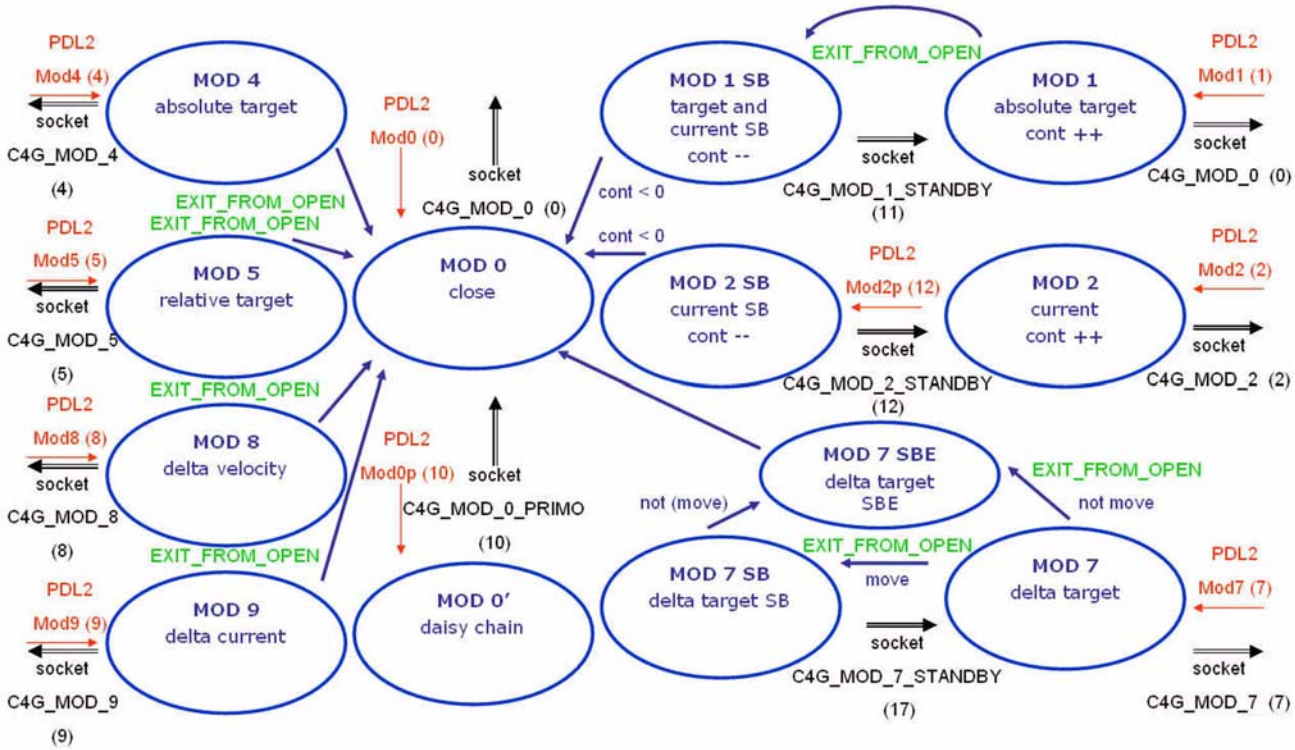


Fig. 24.3 - Actions performed by SW_PC

switch(sx_C4GOpen_Packet_Rx.AX[0].SM)

- case C4G_MOD_0 (0): sx_C4GOpen_Packet_Tx.AX[AX_OPEN].SM = C4G_MOD_0
  - packet replying;
- case C4G_MOD_0_PRIMO (10): sx_C4GOpen_Packet_Tx.AX[AX_OPEN].SM = C4G_MOD_0_PRIMO
  - packet replying;
- case C4G_MOD_1 (1): sx_C4GOpen_Packet_Tx.AX[AX_OPEN].SM = C4G_MOD_1
  - at the first step, states and control parameters are initialized,
  - in a first phase a void trajectory is generated and controlled,
  - later on, a void trajectory is generated and controlled, and the following error is checked, as last step, the C4G_EXIT_FROM_OPEN message is created, a void trajectory is generated and controlled
- case C4G_MOD_1_STANDBY (11): sx_C4GOpen_Packet_Tx.AX[AX_OPEN].SM = C4G_MOD_1_STANDBY
  - a void trajectory is generated and controlled;
- case C4G_MOD_2 (2): sx_C4GOpen_Packet_Tx.AX[AX_OPEN].SM = C4G_MOD_2
  - at the first step, states and control parameters are initialized,
  - the trajectory is controlled;
- case C4G_MOD_2_STANDBY (12): sx_C4GOpen_Packet_Tx.AX[AX_OPEN].SM = C4G_MOD_2_STANDBY
  - the trajectory is controlled;
- case C4G_MOD_4 (4): sx_C4GOpen_Packet_Tx.AX[AX_OPEN].SM = C4G_MOD_4
  - at the first step, states are initialized,
  - position and velocity references are generated
  - as last step, the C4G_EXIT_FROM_OPEN message is created and the received references are sent
- case C4G_MOD_5 (5): sx_C4GOpen_Packet_Tx.AX[AX_OPEN].SM = C4G_MOD_5
  - at the first step the following errors are initialized,
  - velocity (and position) references are generated related to sx_C4GOpen_Packet_Rx.CMD, and the following error is checked
  - as last step, a series of samples are performed with a fixed reference (the last one), related to sx_C4GOpen_Packet_Rx.CMD,
  - at the end of the samples series, the C4G_EXIT_FROM_OPEN message is created and the received packet is sent back;
- case C4G_MOD_7 (7): sx_C4GOpen_Packet_Tx.AX[AX_OPEN].SM = C4G_MOD_7
  - in the first phase the delta target is supplied,
  - then the C4G_EXIT_FROM_OPEN message is created, supplying a null delta target;
- case C4G_MOD_7_STANDBY (17): sx_C4GOpen_Packet_Tx.AX[AX_OPEN].SM = C4G_MOD_7_STANDBY
  - a null delta target is supplied;
- case C4G_MOD_7_STANDBY_EXIT (170): sx_C4GOpen_Packet_Tx.AX[AX_OPEN].SM = C4G_MOD_7_STANDBY_EXIT
  - null references are supplied
- case C4G_MOD_8 (8): sx_C4GOpen_Packet_Tx.AX[AX_OPEN].SM = C4G_MOD_8
  - at the first step the needed parameters are initialized,
  - in a first phase the delta velocity is supplied,
  - later on, the C4G_EXIT_FROM_OPEN message is created and null references are supplied;
- case C4G_MOD_9 (9): sx_C4GOpen_Packet_Tx.AX[AX_OPEN].SM = C4G_MOD_9
  - at the first step the needed parameters are initialized,
  - in a first phase the delta current is supplied,
  - later on, the C4G_EXIT_FROM_OPEN message is created and null references are supplied;
- default:
  - no actions are performed.

Fig. 24.4 - Commands issued by SW_PDL

```
-- Mode 0
  setta_modalita(mod0)
  MOVE (approach) (ind_stato)
  DELAY time_intermod
-- Mode 0'
  setta_modalita(mod0p)
  MOVE (approach) (ind_stato)
  DELAY time_intermod
-- Mode 1
  setta_modalita(mod1)
  DELAY time_mod1_input   -- initial transitory (enter)
  DELAY time_mod1_change  -- variable saturation
  DELAY time_mod1_output  -- final transitory (exit)
  segnale_go := FALSE
  ENABLE CONDITION[cond_mod0]
  WAIT FOR segnale_go
  DELAY time_intermod
-- Mode 2/2'
  setta_modalita(mod2)
  DELAY time_mod2_input   -- initial transitory (enter)
  DELAY time_mod2_change  -- variable saturation
  DELAY time_mod2_output  -- final transitory (exit)
  DELAY time_premove
  MOVE (either void or meaningful)
  setta_modalita(mod12)
  DELAY time_mod2_input   -- initial transitory (exit)
  DELAY time_mod2_change  -- variable saturation
  DELAY time_mod2_output  -- final transitory (enter)
  DELAY time_intermod
```

```
-- Mode 4
  MOVE (approach)
  DELAY time_premove
  setta_modalita(mod4)
  segnale_go := FALSE
  ENABLE CONDITION[cond_mod0]
  WAIT FOR segnale_go
  DELAY time_intermod
-- Mode 5
  $ARM_DATA[ind_arm].C4GOPEN_CMD[*] := ind_stato
  setta_modalita(mod5)
  segnale_go := FALSE
  ENABLE CONDITION[cond_mod0]
  WAIT FOR segnale_go
  DELAY time_intermod
-- Mode 7
  setta_modalita(mod7)
  segnale_go := FALSE
  ENABLE CONDITION[cond_mod0]
  MOVE (either long or short)
  WAIT FOR segnale_go
  DELAY time_intermod
```

```
-- Mode 8
  setta_modalita(mod8)
  segnale_go := FALSE
  ENABLE CONDITION[cond_mod0]
  MOVE (GENERIC)
  WAIT FOR segnale_go
  DELAY time_intermod
-- Mode 9
  setta_modalita(mod9)
  segnale_go := FALSE
  ENABLE CONDITION[cond_mod0]
  MOVE (GENERIC)
  WAIT FOR segnale_go
  DELAY time_intermod
```

25. SOFTWARE LAYERS

The following software/hardware layers are implemented and handled on the remote PC connected to C4G:

Tab. 25.1 - Software/hardware layers on the remote PC

4	Applications		application program
3	Functions	basic functions	API for functions to be supplied
		FSM	
		C4G OPEN packets	
2	Communication		communication driver
1	Operating System		Linux + RTAI libraries
0	Hardware		--

Since the C4G OPEN is a real-time system, the PC operating system must be real-time too. For such a reason it has been chosen to use Linux **operating system**, compiled with the RTAI real-time processing dedicated libraries.

The UDP/IP **communication** drivers, allowing to exchange packets between SMP+ and PC, must be developped in Linux environment too.

Since a Linux open source distribution is used, the communication libraries are distributed in open modality too.

Over the communication layer there is the **functions** layer.

Such a layer can be divided into three sub-layers; from the highest to the lowest we have:

– **basic functions**,

– **FSM** (Finite States Machine),

– **C4G OPEN packets** handling.

The sub-level handling C4G OPEN communication packets, takes care of creating, sending and receiving PC-side C4G OPEN packets.

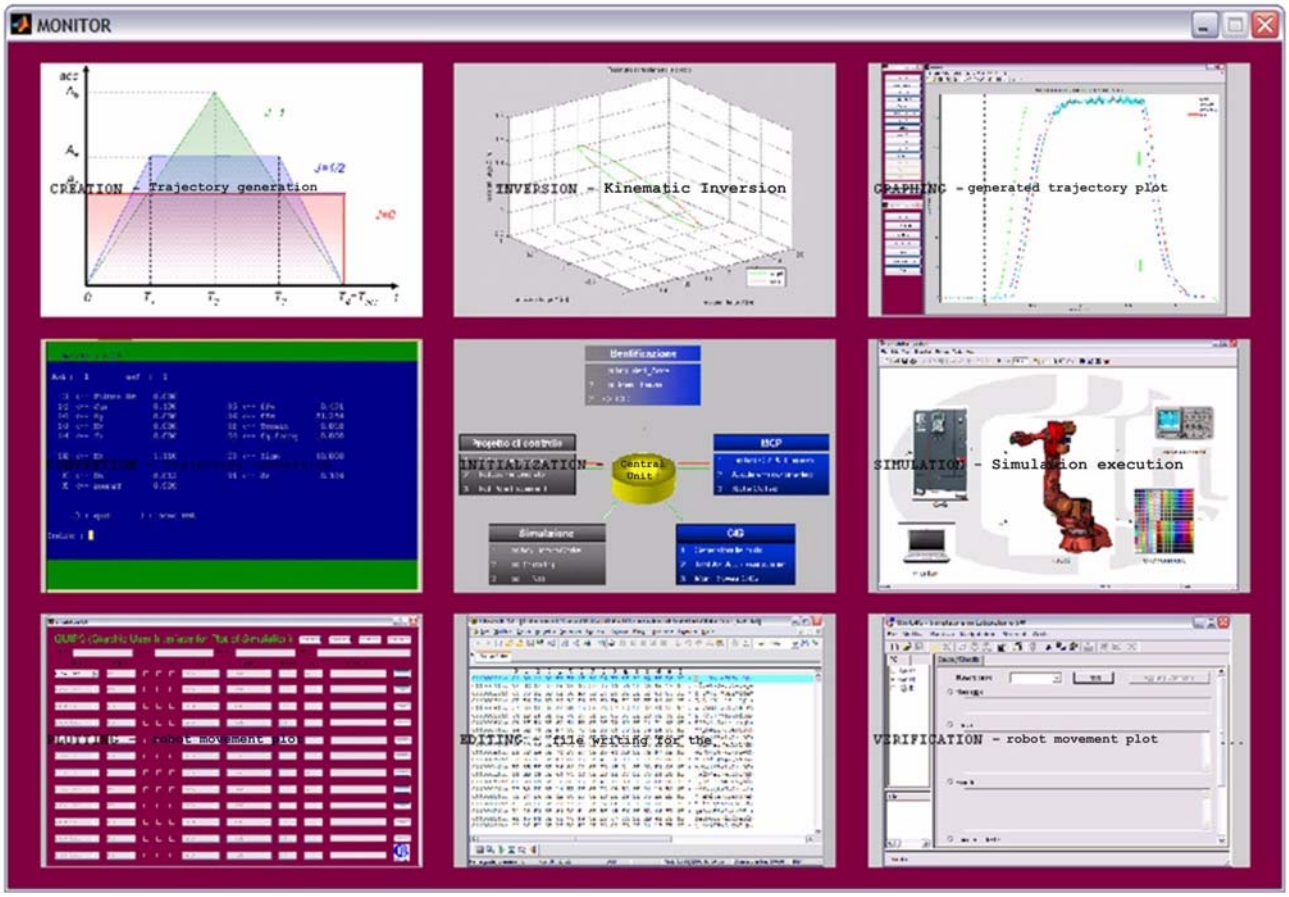
SMP+-side, this sub-level performs the same functions but it also handles BLIP and RAAZ, filtering them and finding/inserting information to be supplied to the C4G OPEN packets towards PC, according to what described in Chap.9.

The FSM, instead, takes care of handling OPEN modalities and changing from an operating modality to another.

The basic functions are several basic primitives to be supplied the upper layers with.

At that level there must be all the procedures to allow axes handling in open modality: they choose the specified open modality type, they schedule the trajectory generation and motion control algorithms to be executed according to the SMP+ requests, they receive information from standard sensors, etc.
This software layer must also include the HOLD procedure from PC, upon SMP+ command, and the following error redundant management on PC.

Fig. 25.1 - Application software example for C4G OPEN



C4G OPEN system project main target is to be able to use (in its most general meaning) a COMAU robot, issuing commands from a PC.

In this meaning, the application level can be developed by any users, according to their own particular needs.
In any case, to check that the complete pile of software layers properly works, a basic application software must be implemented which, even if with restricted functionalities, allows to easily and immediately use the whole system.

Fig. 25.1 shows, as an example, the graphic interface (implemented in MatLab/SimuLink) of an application program using C4G OPEN to control a robot by means of targets which have been previously calculated by a PC.
Such an example is useful to understand which must be the functions accomplished by such an application program:

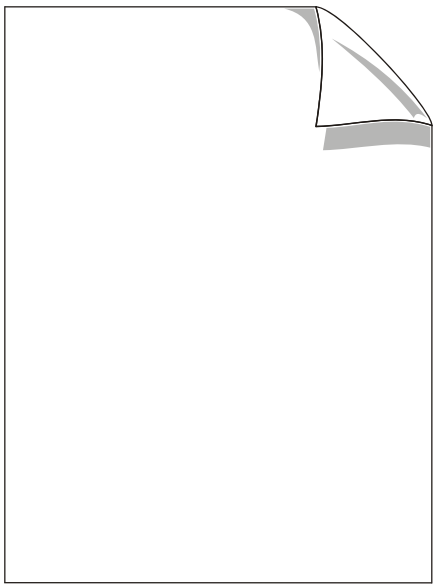
– **trajectory generation**: this module, according to the robot characterization, loads a file containing the movements to be executed by the machine, and generates the most appropriate points sequence to perform the required trajectory;

– **simulation tool**: this module, knowing the robot characterization and the border conditions such as payload on the flange and mounting type (on floor, on ceiling, tilted), simulates the robot behaviour during the trajectory which has been calculated by the trajectory generation module;

– **translation program**: this module translates the previously calculated trajectory from PC format to C4G controller compliant format;

– **command console**: this module monitors what is happening on C4G; mainly, it must supply the following functionalities:

- **displaying**: it allows to monitor the system state, graphically showing the manipulator motion, by means of the actual positions measures coming from the encoders;
- **control**: it allows to let the robot performing easy tasks (such as small joint movements) issuing commands by means of the graphics interface.



26. SOFTWARE ON PC

26.1 Introduction

In the current chapter the following topics are described:

– Fedora distribution

– Ubuntu distribution.

READ CAREFULLY - Besides the C4GOpen software option, the following items are provided:

– **“C4G Open Library for Linux and OrchestraC4GLibrary v1.17.00” library to interface C4GOpen on LINUX platforms,**

– **an explanatory application developped for a SIX Robot SIX (see directory “C4G Open Example”).**

This all is merely distributed as a demonstration and to provide the user with a base for developing further applications.
Comau shall in no way be held liable for any application the user could develop starting from such examples.

NOTE - For any AMD and/or 64 bit chip architecture, the C4G Open libraries cannot work yet, owing to the RTai distribution limitations.
The proper functioning both of C4GOpen and its related libraries is only guaranteed for INTEL Pentium 32 bit chip architecture.

26.2 Fedora distribution

Some notes about the software configuration of external PC communicating with SMP+, are provided in the current paragraphs.

Further details in the following document: **C4G Open Setup HW & SW "PC-side"** (see Tab. 27.1 - Reference documents).

Linux Fedora Core 4 distribution:

– http://fedora.redhat.com/

– http://redhat.download.fedoraproject.org/pub/fedora/linux/core/4/i386/
ISO files to be downloaded and then recorded on 4 CDs:
  • FC4-i386-disc1.iso,
  • FC4-i386-disc2.iso,
  • FC4-i386-disc3.iso,
  • FC4-i386-

Kernel vanilla 2.6.16:

– http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.16.tar.bz2

RTAI 3.4:

– https://www.rtai.org/RTAI/rtai-3.4.tar.bz2
loading script:

```
#!/bin/sh

MOD_DIR=/usr/realtime/modules

sync
make -f /usr/src/rtai/GNUmakefile dev
insmod ${MOD_DIR}/rtai_hal.ko
insmod ${MOD_DIR}/rtai_sched.ko
insmod ${MOD_DIR}/rtai_fifos.ko
insmod ${MOD_DIR}/rtai_sem.ko
insmod ${MOD_DIR}/rtai_mbx.ko
insmod ${MOD_DIR}/rtai_msg.ko
insmod ${MOD_DIR}/rtai_netrpc.ko ThisNode="127.0.0.1"
insmod ${MOD_DIR}/rtai_math.ko
sync
```

RTnet 0.97:

– http://www.rts.uni-hannover.de/rtnet/download/rtnet-0.9.7.tar.bz2
loading script:

```
#!/bin/sh

MOD_DIR=/usr/local/rtnet/modules

sync
mknod /dev/rtnet c 10 240
insmod /usr/realtime/modules/rtai_rtdm.ko
insmod ${MOD_DIR}/rtnet.ko
insmod ${MOD_DIR}/rtipv4.ko
insmod ${MOD_DIR}/rtpacket.ko
insmod ${MOD_DIR}/rt_loopback.ko
insmod ${MOD_DIR}/rt_eepro100.ko
insmod ${MOD_DIR}/rtcap.ko
/usr/local/rtnet/sbin/rtifconfig rtlo up 127.0.0.1
/usr/local/rtnet/sbin/rtifconfig rteth0 up 10.2.12.149
/usr/local/rtnet/sbin/rtroute solicit 10.2.12.150 dev rteth0
sync
```

Compiler:

```
[c4gopen@localhost ~]$ gcc -v
Using built-in specs.
Target: i386-redhat-linux
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man
--infodir=/usr/share/info --enable-shared --enable-threads=posix
--enable-checking=release --with-system-zlib --enable-__cxa_atexit
--disable-libunwind-exceptions --enable-libgcj-multifile
--enable-languages=c,c++,objc,java,f95,ada --enable-java-awt=gtk
--with-java-home=/usr/lib/jvm/java-1.4.2-gcj-1.4.2.0/jre --host=i386-redhat-linux
Thread model: posix
gcc version 4.0.0 20050519 (Red Hat 4.0.0-8)
```

Start message on the screen: Fedora Core 2.6.11-1.1369_FC4

End message on the screen: Fedora Core 2.6.16-rtai.

26.3 Ubuntu distribution

The following topics are described about this distribution:

- Software requirements
- Hardware requirements
- Compliant network cards
- Kernel installation
- RTai module loading
- RTNet module loading
- C4G Open application execution.

26.3.1 Software requirements

- **Ubuntu 7.04 "The Feisty Fawn" CD x86 edition**: Ubuntu Website - http://www.ubuntu.com
- **Linux vanilla kernel 2.6.19**: The Linux Kernel Archives - http://www.kernel.org
- **RTAI 3.5**: RTAI Website - http://www.rtai.org
- **RTnet 0.9.9**: RTnet Website - http://www.rtnet.org
- **GNU g++ 4.1.2**: The GNU Project - http://www.gnu.org
- **GNU bash 3.2.13(1)**: The GNU Project - http://www.gnu.org

Start message on the screen: Ubuntu Kernel 2.6.20_15_generic
End message on the screen: Ubuntu Kernel 2.6.19_rtai_generic.

26.3.2 Hardware requirements

- **One x86-compatibile PC** (at least Pentium chip set class)
- **32 bit architecture** ONLY
- **Three UTP cat.5 or higher network cables**
- **One Fast Ethernet switch**
- **One RTnet-compliant network card** (10/100 Mbps) (see par. 26.3.3 Compliant network cards on page 26-3).

26.3.3 Compliant network cards

For C4G Open to work properly, the network card must be setup for the hard real-time communication between external PC and C4G.

To understand whether a network card allows the hard real-time communication or not, Ubuntu distribution provides information about the NIC, included the chipset model and the IRQ by means of the following command

```
lspci -v
```

In the below figure, an example is shown of the command output for a network card tested by *Sintesi Scpa* - Bari (Italy) - cooperating with COMAU for the PC software development, in the C4G OPEN project:

```
00:0b.0 Ethernet controller: Digital Equipment Corporation DECchip 21140 [FasterNet] (rev 22)
        Subsystem: D-Link System Inc DFE-500TX Fast Ethernet
        Flags: bus master, medium devsel, latency 32, IRQ 10
        I/O ports at c800 [size=128]
        Memory at e8143000 (32-bit, non-prefetchable) [size=128]
        [virtual] Expansion ROM at 30000000 [disabled] [size=256K]
```

It is possible to get further information about the hardware configuration, by means of *hwinfo* utility, available in the *Synaptic Package Manager* software package (which can be called by means of *sudo synaptic* command).

Moreover, to get information about any installed PCI devices, it could be useful to issue the following command

```
hwinfo --pci
```

NOTE THAT if the shown above command doesn't work properly, the user should execute step C (see par. 26.3.4 Kernel installation on page 26-8).

In the following figure, the command output related to a tested hardware, is shown.

Please note that information about the standard Linux driver name for the different devices is provided too:

```
19: PCI 0b.0: 0200 Ethernet controller
  [Created at pci.281]
  UDI: /org/freedesktop/Hal/devices/pci_1011_9
  Unique ID: JNkJ.AujCtKsDPG3
  SysFS ID: /devices/pci0000:00/0000:00:0b.0
  SysFS BusID: 0000:00:0b.0
  Hardware Class: network
  Model: "D-Link DFE-500TX Fast Ethernet"
  Vendor: pci 0x1011 "Digital Equipment Corporation"
  Device: pci 0x0009 "DECchip 21140 [FasterNet]"
  SubVendor: pci 0x1186 "D-Link System Inc"
  SubDevice: pci 0x1100 "DFE-500TX Fast Ethernet"
  Revision: 0x22
  Driver: "tulip"
  Driver Modules: "tulip"
  Device File: eth1
  I/O Ports: 0xc800-0xc87f (rw)
  Memory Range: 0xe8143000-0xe814307f (rw,non-prefetchable)
  Memory Range: 0x30000000-0x3003ffff (ro,prefetchable,disabled)
  IRQ: 10 (8 events)
  HW Address: 00:40:05:37:36:e4
  Module Alias: "pci:v00001011d00000009sv00001186sd00001100bc02sc00i00"
  Driver Info #0:
    Driver Status: tulip is active
    Driver Activation Cmd: "modprobe tulip"
  Config Status: cfg=new, avail=yes, need=no, active=unknown
```

It is important to be sure that the IRQ used by the NIC chipset is not shared with any other devices, in order not to compromise the system hard real-time communication. To check that, issue *cat /proc/interrupts* command.

In case of IRQ sharing, it is needed to either modify the network card PCI slot, or to disable the other devices, or to remove the kernel modules associated to them.

In the following figure, the *cat* command output, related to the tested hardware, is shown:

```
 0:     291251   XT-PIC-XT   timer
 1:          2   XT-PIC-XT   i8042
 2:          0   XT-PIC-XT   cascade
 3:        119   XT-PIC-XT   SiS SI7012
 5:      16674   XT-PIC-XT   eth0
 6:          5   XT-PIC-XT   floppy
 7:          1   XT-PIC-XT   parport0
 8:          3   XT-PIC-XT   rtc
 9:          0   XT-PIC-XT   ehci_hcd:usb2
10:          8   XT-PIC-XT   ohci_hcd:usb1, eth1
11:      67755   XT-PIC-XT   ohci_hcd:usb3, radeon@pci:0000:01:00.0
14:      27017   XT-PIC-XT   ide0
```

For example, if the network card shares IRQ 10 channel with *ohci_hcd* module, it is necessary to remove it, issuing the following command either from ROOT or in SUDO modality:

```
rmmod ohci_hcd
```

It would be better to use PS/2 mouse and keyboard, in order to prevent USB devices to conflict with the card IRQ channel.

The following table shows the list of hard real-time RTNet-compliant cards, at the time in which this document is written:

– Tab. 26.1 - Hard real-time RTNet-compliant cards (1)

– Tab. 26.2 - Hard real-time RTNet-compliant cards (2)

Tab. 26.1 - Hard real-time RTNet-compliant cards (1)

RTnet driver name	Vendor ID	Device ID	Manufacturer	Chip number
rt_tulip	0x1011	0x0002	DEC	DC21040
	0x1011	0x0014	DEC	DC21041
	0x1011	0x0009	DEC	DC21140
	0x1011	0x0019	DEC	DC21142/3
	0x11ad	0x0002	Lite-On Communications	NGMC169B
	0x11ad	0xc115	Lite-On Communications	LC82C115
	0x10d9	0x0512	Macronix	MX98713
	0x10d9	0x0531	Macronix	MX98715/25
	0x1317	0x0981	ADMtek	AN981
	0x1317	0x0985	ADMtek	AN983B
	0x1317	0x1985	ADMtek	AN985
	0x1317	0x9511	ADMtek	ADM9511
	0x104a	0x0981	STMicroelectronics	21x4x
	0x104a	0x2774	STMicroelectronics	21x4x
	0x11f6	0x9881	Powermatic	TXA9881
	0x1282	0x9102	Davicom	DM9102/A/AF
	0x1113	0x1216	Accton	EN5251BE
	0x1113	0x1217	Accton	EN2242
	0x1113	0x9511	Accton	SMC EN5251BE
rt_3c59x (experimental)	0x10b7	0x5900	3Com	3C590
	0x10b7	0x5920	3Com	3C592
	0x10b7	0x5970	3Com	3C597
	0x10b7	0x5950	3Com	3C595
	0x10b7	0x5951	3Com	3C595
	0x10b7	0x5952	3Com	3C595
	0x10b7	0x9000	3Com	3C900
	0x10b7	0x9001	3Com	3C900
	0x10b7	0x9004	3Com	3C900B-TPO
	0x10b7	0x9005	3Com	3C900B-Combo
	0x10b7	0x9006	3Com	3C900B-TPC
	0x10b7	0x900a	3Com	3C900B-FL
	0x10b7	0x9050	3Com	3C905
	0x10b7	0x9051	3Com	3C905
	0x10b7	0x9055	3Com	3C905B
	0x10b7	0x9058	3Com	3C905B
	0x10b7	0x905a	3Com	3C905B-FX
	0x10b7	0x9200	3Com	3C905C-TX/TX-M
	0x10b7	0x9800	3Com	3C980-TX
	0x10b7	0x9805	3Com	3C980-C
	0x10b7	0x7646	3Com	3CSOHO100-TX
	0x10b7	0x5055	3Com	3C555
	0x10b7	0x6055	3Com	3C556
	0x10b7	0x6056	3Com	3C556B
	0x10b7	0x5b57	3Com	3C595
	0x10b7	0x5057	3Com	3C575
	0x10b7	0x5157	3Com	3CCFE575BT
	0x10b7	0x5257	3Com	3CCFE575CT
	0x10b7	0x6560	3Com	3CCFE656
	0x10b7	0x6562	3Com	3CCFEM656B
	0x10b7	0x6564	3Com	3CXFEM656C
	0x10b7	0x4500	3Com	3C450

Tab. 26.2 - Hard real-time RTNet-compliant cards (2)

RTnet driver name	Vendor ID	Device ID	Manufacturer	Chip number
rt_8139too	0x10ec	0x8139	RealTek	RTL8139/8139C/8139C+
	0x10ec	0x8138	RealTek	RTL8139 (B/C)
	0x4033	0x1360	Delta Networks	RTL8139
	0x1186	0x1300	D-Link	RTL8139
	0x1186	0x1340	D-Link	DFE-690TXD
	0x13d1	0xab06	AboCom	RTL8139
	0x1259	0xa117	Allied Tsyn	RTL81xx
rt_eepro100	0x8086	0x1229	Intel	82557/8/9/0/1
	0x8086	0x1209	Intel	8255xER/IT
	0x8086	0x1029	Intel	82559
	0x8086	0x1030	Intel	825593
	0x8086	0x1031	Intel	82801CAM
	0x8086	0x1032	Intel	82801CAM
	0x8086	0x1033	Intel	82801CAM
	0x8086	0x1034	Intel	82801CAM
	0x8086	0x1035	Intel	82562EH
	0x8086	0x1036	Intel	82562EH
	0x8086	0x1037	Intel	82801CAM
	0x8086	0x1038	Intel	82801CAM
	0x8086	0x1039	Intel	82801DB
	0x8086	0x103A	Intel	82801DB
	0x8086	0x103B	Intel	82801DB
	0x8086	0x103C	Intel	82801DB
	0x8086	0x103D	Intel	82801DB
	0x8086	0x103E	Intel	82801DB
	0x8086	0x1227	Intel	82865g
	0x8086	0x1228	Intel	82556
	0x8086	0x2449	Intel	82559ER
	0x8086	0x2459	Intel	82801E
	0x8086	0x245D	Intel	82801E
	0x8086	0x5200	Intel	EE PRO/100
rt_natsemi	0x100b	0x0001	National Semiconductors	DP83810
	0x100b	0x0020	National Semiconductors	DP83815/16

26.3.4 Kernel installation

After the Ubuntu 7.04 version has been installed, the Kernel installation is to be performed as follows:

a. **step A**
- sudo dpkg -i linux-headers-2.6.19-rtai-generic_r1_i386.deb (RTAI Kernel)
- sudo dpkg -i linux-source-2.6.19-rtai-generic_r1_all.deb (RTAI Kernel Source)
- sudo dpkg -i linux-image-2.6.19-rtai-generic_r1_i386.deb (RTAI Kernel)
- sudo dpkg -i libfreetype6_2.2.1-5ubuntu1.1_i386.deb
- sudo dpkg -i libgl1-mesa-glx_6.5.2-3ubuntu8_i386.deb
- sudo dpkg -i libgl1-mesa-dri_6.5.2-3ubuntu8_i386.deb
- sudo dpkg -i libglu1-mesa_6.5.2-3ubuntu8_i386.deb

b. **step B**
- tar xjvf rtai-base_1.0-3_i386_full.tar.bz2
- sudo dpkg -i rtai-base_1.0-3_i386_full/*.deb (RTAI Base)

c. **step C**
- sudo dpkg -i libhd13_13.11-3_i386.deb (Informazioni dell'hardware)
- sudo dpkg -i hwinfo_13.11-3_i386.deb (Informazioni dell'hardware)

d. **step D**
   – sudo reboot (choosing ubuntu-rtai)

e. **step E**
   – cd /usr/src/rtai-base
   – sudo make
   – sudo make upgrade

26.3.5 RTai module loading

The **first step to use C4G Open** is to load RTai module, to provide the computer with real-time processing skill.
The loadRTAI script (shown here below) which is in ***/usr/local/C4Gopen/bin*** directory, must be executed either from ROOT or in SUDO modality: ./loadRTAI

```
#!/bin/bash

RTAI_SRC_DIR=/usr/src/rtai
RTAI_MODULES_DIR=/usr/realtime/modules

sync
make -f ${RTAI_SRC_DIR}/GNUmakefile dev
insmod ${RTAI_MODULES_DIR}/rtai_hal.ko
insmod ${RTAI_MODULES_DIR}/rtai_lxrt.ko
insmod ${RTAI_MODULES_DIR}/rtai_fifos.ko
insmod ${RTAI_MODULES_DIR}/rtai_sem.ko
insmod ${RTAI_MODULES_DIR}/rtai_mbx.ko
insmod ${RTAI_MODULES_DIR}/rtai_msg.ko
insmod ${RTAI_MODULES_DIR}/rtai_rtdm.ko
insmod ${RTAI_MODULES_DIR}/rtai_netrpc.k
ThisNode="127.0.0.1"
sync
```

Please note that, at EACH computer restart, RTAI module must ALWAYS be loaded.

26.3.6 RTNet module loading

In order to **provide PC with communication hard real-time features**, the RTNet kernel modules must be loaded and the subnet for C4G Open must be configured.

The loadRTnet script which is in ***/usr/local/C4Gopen/bin*** directory must be replaced with the here below instructions, taking care of writing the right name of RTNet and Linux drivers (examples written in red).

To properly use the listed below instructions, carefully read the NOTEs:

```
#!/bin/bash

RTAI_MOD_DIR=/usr/realtime/modules
RTNET_MOD_DIR=/usr/local/rtnet/modules
RTNET_BIN_DIR=/usr/local/rtnet/sbin
```

```
ETH_DRV=linux_driver_name  (e.g. 8139too <- to remove (*))
RT_ETH_DRV=rtnet_driver_name(e.g. rt_8139too <- to remove(*))

LOCAL_IP=10.2.12.149
TARGET_IP=10.2.12.150

mknod /dev/rtnet c 10 240

rmmod ${ETH_DRV}

insmod ${RTNET_MOD_DIR}/rtnet.ko
insmod ${RTNET_MOD_DIR}/rtipv4.ko
insmod ${RTNET_MOD_DIR}/rtpacket.ko
insmod ${RTNET_MOD_DIR}/rt_loopback.ko
insmod ${RTNET_MOD_DIR}/${RT_ETH_DRV}.ko

${RTNET_BIN_DIR}/rtifconfig rtlo up 127.0.0.1
${RTNET_BIN_DIR}/rtifconfig rteth0 up ${LOCAL_IP}

${RTNET_BIN_DIR}/rtroute solicit ${TARGET_IP} dev rteth0
```

NOTEs

- **(*) - please, delete the example in red, before using the listed above instructions**
- **note that ETH_DRV variable is referred to the module name of the network card driver kernel, whilst the RT_ETH_DRV variable is referred to the module name of the network card hard real-time driver RTnet kernel.**
  Please, see par. 26.3.2 Hardware requirements on page 26-3 to get information about the installed NIC and to know the name of the network card driver.
  As far as the RTNet driver name, refer to previous Tab. 26.1 - Hard real-time RTNet-compliant cards (1) and Tab. 26.2 - Hard real-time RTNet-compliant cards (2).
  E.g., if the chipset is RTL8139, the Linux standard drive name is 8139too, whilst the RTNet driver is rt_8139too; so the needed instructions are:

  ```
  ETH_DRV=8139too
  RT_ETH_DRV=rt_8139too
  ```

- **note that the value of the LOCAL_IP variable is "10.2.12.149", while the value of the TARGET_IP variable is "10.2.12.150"; they are, respectively, the PC IP address and the SMP+ IP address, within the C4G Open subnet.**

After the script has been modified, it is to be saved and then to be executed issuing the following command in ***/usr/local/C4Gopen/bin*** directory, either from ROOT or SUDO modality:

```
./loadRTnet
```

At this point, PC is ready to communicate with C4G Open.

Please note that at EACH computer restart, RTNet module must ALWAYS be executed.

26.3.7 C4G Open application execution

a. Unzip Socket_C4G_Open.zip into a read/write attributes directory.

b. Set the directory and compile the code by means of the following command:

```
make
```

c. Check the compilation properly terminates (no errors), then execute the application, by means of the following command:

```
sudo ./socketC4Gopen
```

d. Then the C4GOpen software is running, waiting for the first message to come from C4G.

e. As soon as it comes, the initialization information is displayed as shown in Fig. 10.1. Since now and then, the system is ready to switch to one of the open modalities, as intended by the required application.

Note that the software running on external PC acts as a server, so it must start before C4G starts for the first time, after having set the required variables to enable axes to the open modality, according to what is described in Chap.6. - System initial configuration.

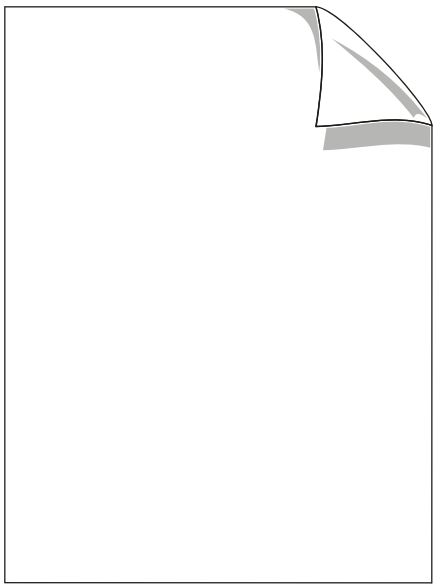
The code compilation must be performed EVERY TIME the program is modified.

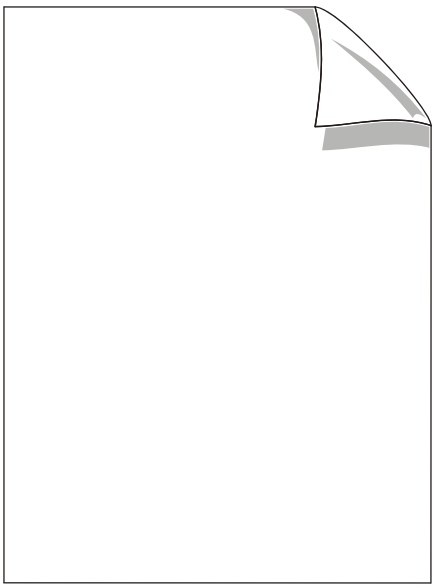
27. REFERENCE DOCUMENTS

Tab. 27.1 lists the C4G OPEN reference documents.

Tab. 27.1 - Reference documents

N°	Title
1	C4G Open Setup HW & SW "PC-side"
2	Troubleshooting RTAI & COMEDI





COMAU

Comau in the World

COMAU S.p.A.
Headquarters
Via Rivalta, 30
10095 Grugliasco - TO (Italy)
Tel. +39-011-0049111

Powertrain Machining & Assembly
Via Rivalta, 30-49
10095 Grugliasco - TO (Italy)
Tel. +39-011-0049111
Telefax +39-011-0049688

Body Welding & Assembly
Strada Borgaretto, 22
10092 Borgaretto di Beinasco - TO (Italy)
Tel. +39-011-0049111
Telefax +39-011-0048672

Robotics & Service
Via Rivalta, 30
10095 Grugliasco - TO (Italy)
Tel. +39-011-0049111
Telefax +39-011-0049866

Engineering, Injection Moulds & Dies
Via Bistagno, 10
10136 Torino (Italy)
Tel. +39-011-005171
Telefax +39-011-0051882

Comau France S.A.
5-7, rue Albert Einstein
78197 Trappes Cedex (France)
Tel. +33-1-30166100
Telefax +33-1-30166209

Comau Estil
10, Midland Road
Luton, Bedfordshire LU2 0HR (UK)
Tel. +44-1582-817600
Telefax +44-1582-817700

Comau Deutschland GmbH
Monzastrasse 4D
D-63225 Langen (Germany)
Tel. +49-6103-31035 -0
Telefax +49-6103-31035-29

German Intec GmbH & Co. KG
Im Riedgrund 1
74078 Heilbronn (Germany)
Tel. +49-7131 28 22-0
Telefax +49-731 28 22-400

Mecaner S.A.
Calle Aita Gotzon 37
48610 Urduliz - Vizcaya (Spain)
Tel. +34-94-6769100
Telefax +34-94-6769132

Comau Poland Sp. ,Z.O.O.
Ul. Turyńska 100
43-100 Tychy (Poland)
Tel. +48-32-2179404
Telefax +48-32-2179440

Comau Romania S.R.L.
Oradea, 3700 Bihor
Str. Berzei nr.5 Suite E (Romania)
Tel. +40-59-414759
Telefax +40-59-479840

Comau Russia S.R.L.
Ul. Bolshaya Dmitrovka 32/4
107031 Moscow (Russian Federation)
Tel. +7-495-7885265
Telefax +7-495-7885266

Comau SPA Turkiye Bursa Isyeri
Panayir Mah. Buttimis Is Merkezi
C Block Kat 5 no.1494
16250 Osmangazi/Bursa (Turkey)
Tel. +90-0224-2112873
Telefax +90-0224-2112834

Comau Inc.
21000 Telegraph Road
Southfield, MI 48034 (USA)
Tel. +1-248-3538888
Telefax +1-248-3682531

Comau Pico Mexico S. de R.L. de C.V.
Av. Acceso Lotes 12 y 13
Col. Fracc. Ind. El Trébol 2° Secc.
C.P. 54610, Tepotzotlan (Mexico)
Tel. +11-52-5 8760644
Telefax +11-52-5 8761837

Comau Canada Inc.
4325 Division Road Unit # 15
Ontario N9A 6J3 (Canada)
Tel. +1-519-9727535
Telefax +1-519-9720809

Comau do Brasil Ind. e Com. Ltda.
Rua Do Paraíso, 148 - 4° Andar
Paraíso - Cep. 04103-000
São Paulo - SP (Brazil)
Tel. +55-11-21262424
Telefax +55-11-32668799

Comau Argentina S.A.
Ruta 9, Km 695
5020 - Ferreyra
Córdoba (Argentina)
Tel. +54-351-4503996
Telefax +54-351-4503909

Comau SA Body Systems (Pty)
Hendrik van Eck Drive
Riverside Industrial Area
Uitenhage 6229 (South Africa)
Tel. +27-41-9953600
Telefax +27-41-9229652

Comau (Shanghai) Automotive Equipment Co., Ltd.
Pudong, Kang Qiao Dong Road Nr. 1300
Block 2 — Kang Qiao
201319 Shanghai (P.R.China)
Tel. +86-21-68139900
Telefax +86-21-68139622

Comau India Pvt. Ltd.
33Km Milestone Pune-Nagar Road
Shikrapur, Pune - 412208 (India)
Tel. +91.2137.678100
Telefax +91.2137.678110

COMAU Robotics services
Repair: **repairs.robotics@comau.com**
Training: **training.robotics@comau.com**
Spare parts: **spares.robotics@comau.com**
Technical service: **service.robotics@comau.com**
comau.com/robotics