Proceeding of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space: i-SAIRAS 2003, NARA, Japan, May 19-23, 2003

EUROPA Robot Controller: advances in design

A. Olivieri (Agenzia Spaziale Italiana, Matera, Italy) e-mail angelo.olivieri@asi.it
T. Grasso (Tecnomare, Venice, Italy) e-mail: grasso.t@tecnomare.it
A. Rusconi (Tecnospazio, Milan, Italy) e-mail: arusconi@tecnospazio.it
M. Colomba (Tecnospazio, Milan, Italy) e-mail: mcolomba@tecnospazio.it

Keywords space robot controller, force control, redundancy, NASREM, manipulation, safety barriers

Abstract

This paper describes the main characteristics and the advances in design of the EUROPA robot controller, presented in [1].
EUROPA (External Use of Robotics for Payloads Automation) is a robotic experiment for externally exposed payloads, which will be installed on an EXPRESS Pallet Adapter of the International Space Station.

1. Introduction

The EUROPA robot controller will be an externally exposed unit (as part of the EUROPA flight segment items mounted on the EXPRESS Pallet), fully dedicated to controlling the robot arm, to provide the necessary computing power to include advanced control algorithms and reach a high degree of autonomy for the on-board operations.
The EUROPA robot controller provides the following features:

- execution, in nominal mode, of the robotic programs written using the EUROPA Programming Language (EPL);
- control of a robot arm with seven joints and an end effector (a gripper with two parallel jaws), with real-time management of the arm kinematics redundancy; in emergency mode only one joint at a time is moved;
- motion control capabilities in free space, using internal sensors (resolvers);
- contact motion control capabilities (using force/torque sensor), only in nominal mode;
- modular software architecture based on hierarchical NASREM concept;
- implementation of a computer based control system fail safe approach, such that the overall EUROPA system will be two-fail safe for specific hazards related to the robot manipulator, which is a moving object (arm outside of defined workspace and collision effects);
- achievement of a good degree of failure tolerance, in order to guarantee, in front of one failure, safe stowage of the robot arm and therefore to minimize astronaut Extra Vehicular Activity (EVA) intervention; this is mainly obtained with the use of nominal and emergency controllers.

This paper is organized as follows.
Section 2 contains a general description of the EUROPA system, as for the current design.
Section 3 is dedicated to the EUROPA robot controller.
Section 4 contains the conclusion.

2. EUROPA Flight System Configuration

The EUROPA flight segment (for a more detailed description, see [5]) is divided into four subsystems:

- Robotic subsystem
- Facility subsystem
- Science and technology subsystem
- Crew MMI subsystem.

The external part of the EUROPA flight segment (Robotic subsystem, Facility subsystem, Science and technology subsystem) will be accommodated on the EXPRESS Pallet Adapter (ExPA). The Crew MMI subsystem will be a Portable Computer System (PCS) available to the crew inside the Space Station.

The accommodation of EUROPA flight segment on the ExPA is shown in Fig. 1 (picture taken from EUROPA mock-up).

Fig. 1 EUROPA Flight Segment Accommodation

3. EUROPA Robot Controller

3.1 Overview

EUROPA robot controller is constituted by two alternate systems (see [1] for details):

- the nominal controller, used to demonstrate the capability of a robotic arm to perform specific operations, and scientific experiment management, in both free motion and contact motion;
- the emergency controller, used in case of failure of the nominal controller HW/SW, to stow the arm in the suitable latch devices to be ready for the re-entry

Nominal controller is equipped with a 6U VME crate, containing the CPU board (Aitech S210, MPC750 processor based), and custom boards for the I/O, 8 motor drivers, motor resolvers acquisition, power supply, output shaft resolver acquisition, two microcontrollers for safety

Emergency controller is equipped with the CPU board (Astrium SPLC, Sparc V7-ERC32 processor based), and custom boards for the I/O and power supply. The boards for 8 motor drivers, motor resolvers acquisition, output shaft resolver acquisition, one microcontroller for safety are the same as in nominal mode. Boards are connected through RS485 bus.

In both nominal and emergency modes, microcontroller boards (8086 processor based) will be used to guarantee the foreseen level for safety checks.

The software related to both nominal and emergency modes is developed using C++ language and Rational Rose RT modeler and code generator. This is a UML based tool that allow developers to model the entire software according to some basic components:

- classes, representing the single objects of a system; they are the same classes as in C++ language
- capsules, representing independent flows of control in a system. Capsules have much of the same properties as classes; for example, they can have operations and attributes. Capsules may also participate in dependency, generalization, and association relationships. However, they also have several specialized properties that distinguish them from classes. In particular: external interfaces are only public ports and messaging using suitable protocols, attributes are always private, behaviour is defined by a set of state machines ;
- ports, that are objects whose purpose is to send and receive messages to and from capsule instances
- protocols, that represent the set of messages exchanged between two objects conforms to some communication pattern

Capsules and state machines can be nested, for a better description of the behaviour of the system.

3.2 Nominal Controller Features

From the user's point of view, the nominal controller shall be able to feature the following main characteristics:

- Arm motion in free space
- Arm motion in contact with the environment
- Possibility to command the arm using a high level programming language
- Reception of telecommands and telemetry generation
- Hazards control
- Checks to monitor hardware items

In order to implement them, a suitable number of main capsules have been introduced. In particular, one capsule (Robotic System Controller Capsule) is related to the control of the robotic arm .

Robotic System Controller architecture will be based on hierarchical NASREM concept, structured in Task Level and E-Move Level (for Robotic Programs and single motion instructions decomposition) and Primitive Level and Servo Level (for trajectory definition and servo control). This hierarchical decomposition of the system functions is supported by a horizontal decomposition describing the typology of the functions for a given level. There will be *Task Decomposition*, in which robot motion is decomposed according to the hierarchical level, *Sensory Processing* where all the sensed data are suitably processed, and *World Modeling* that provides suitable world reconstruction and modeling algorithms. The NASREM architecture is reported in Fig. 2.

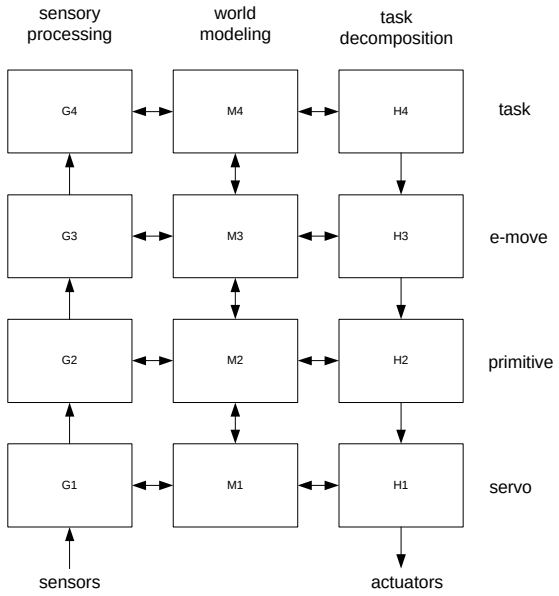

Fig. 2 NASREM architecture

Not shown in the figure for simplicity there are the *Global Memory*, that is the entire database of the system, written only by *World Modeling*, and the *Supervisor*. These two modules can communicate with each module of the figure.
After the early testing and Software Requirement phases, the NASREM levels are well identified and described in the next paragraphs.

3.2.1 Robotic Controller Task Level

For the Robotic controller, a "task" is composed by a Robotic Program (RP), written by a human operator using a high level language (EUROPA Programming Language, EPL). Supervisor has in charge of loading the RP into execution memory, performing also a syntax checking and translation into a suitable binary internal format. The motion planner inside H4 mainly consists of the language interpreter, that takes the internal format file and interprets it "line-by-line".
Task level manages the RP execution according to the state machine of Fig. 3.

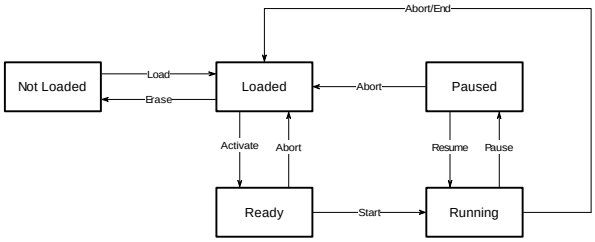

Fig. 3 RP execution

Up to two RP can be executed, once provided that the first RP is not in ready or running state. Further, the second RP cannot be paused.
The main features of EPL are the following:

- Each EPL routine is a PROGRAM, and can be executed as a stand alone RP. The PROGRAM scope is always global.
- Allowed data type are: INTEGER, CHAR, REAL, BOOLEAN, ARRAY (max 3 entries), HPF_DATA. It is not possible to define custom data type.
- Routines can return values of the above data type
- HPF_DATA is a fix structure able to contain all the parameters related to position/force control.
- Variables, of the types above, are allowed. They have always local scope to the routine that define them.
- Expressions evaluation is allowed, for any number of operands.
- Logical and mathematical operands are allowed. A rich math library is provided
- Execution control is based on the following control structures: IF-THEN-ELSE-ENDIF, WHILE-DO-ENDWHILE, DELAY. Nesting is allowed
- Built-in functions, as libraries, for management of data of the robotic cell, I/O, communication with external devices
- Calls to Motion E-Moves

The M4 block is in charge of updating the controller database on the Global Memory.
The G4 block is not implemented.

3.2.2 Robotic Controller E-Move Level

The E-Move level receives E-Move instructions from task level (H4 block). They are the coded form of the following high level instructions:

- MOVE_RELATIVE, for relative linear free motion of the end effector
- MOVE_LINEAR, for linear free motion to a given cartesian setpoint, while keeping the arm angle (see servo level) still
- MOVE_JOINTS, for arm motion in joint space
- MOVE_ANGLE, for motion of the arm angle while keeping the position and orientation of the end effector still.
- MOVE_ALL, for linear free motion to a given cartesian/arm angle setpoint,
- MOVE_AXIS, to move one joint
- MOVE_IN_CONTACT, to move the arm according to the position/force strategy (see servo level)
- MOVE_FROM_CONTACT, to terminate the arm contact motion
- MOVE_TO_CONTACT, to start the arm contact motion
- CHANGE_GRIPPER_WIDTH, to change the opening of the gripper while controlling the width position
- CHANGE_GRIPPER_CURRENT, to change the current to the gripper motor. This ensures, in good approximation, a given gripping strength

Motion finishes according to well defined termination conditions. For free motion, the termination condition is time. For contact motion, the termination conditions can be of four types: reached force setpoint, reached position along a force-controlled degree of freedom, elapsed time, external event. They can be put together in OR mode.
The motion to the given setpoint is considered as an E-Move, that is decomposed into some motion "primitives". Primitives are related to the initial acceleration phase, linear speed phase, deceleration phase, and a "maintain" phase in case of force control. The motion planner inside H3 computes one 5-order

polynomial for each primitive, using suitable boundary conditions on position, speed and acceleration.
The motion decomposition into polynomials is true also for gripper.
The M3 block is in charge of updating the reference frames where motion is defined.
The G3 block is not implemented.

3.2.3 Robotic Controller Primitive Level

The H2 block receives the motion primitives from H3 and samples the polynomials synchronously. The frequency at which H2 runs is 20 Hz. Each polynomial sample is the servo setpoint for the H1 block. At this level, some main items that are used by servo level are computed. They are

- Estimation of the arm dynamics parameters, in terms of inertia matrix, velocity-dependant terms.. All the items are reported to the end effector space (M2)
  They are:
  $$\Lambda = (JA^{-1}J^T)^{-1}$$
  $$m = \bar{J}^T b - \Lambda \dot{J}\dot{q}$$
  where J is the Jacobian, A is the inertia matrix in joint space, b is velocity-dependant term in joint space, $\bar{J}$ is the dynamically consistent inverse of the Jacobian. In this case the arm dynamics equation at end effector space is:
  $$\Lambda \ddot{x} + m(x, \dot{x}) = F - F_C$$
  where x is the end effector position, F is the end effector force and F_C is the commanded force.
- Arm Jacobian (M2)
- Task reference frame computation (M2)
- Automatic computation of the arm angle, for redundancy resolution, based on local repulsion between elbow point and surrounding obstacles. Automatic computation can be overridden if RP requires a direct arm angle setpoint (e.g. operator decides which arm angle to use).

The G2 block is not implemented.

3.2.4 Robotic Controller Servo Level

This level receives from the H2 block the gross setpoints, and operates a fine interpolation at 500 Hz to provide the correct setpoints to the servo loops. Two kind of servo loops are foreseen:

- One cartesian loop or, in alternative, 7 decoupled joint loops
- One loop for gripper position control or, in alternative, one current feed-forward to gripper motor.

The cartesian loop is quite complex and really advanced with respect to the state-of-the-art control schemes. For main concepts, see [1]. The work principle of the control scheme is here reported. Its setpoints are: cartesian task frame position, desired force/moment at task frame, the axes to be controlled in position mode and those controlled in force mode, the arm angle setpoint. The error signals are generated, processed using suitable PID controllers, and an equivalent acceleration signal at the task frame is produced. By multiplying this acceleration by the estimated inertia matrix in cartesian space, and sum the result to the estimated velocity-dependant terms in cartesian space, the required force at task frame is obtained. If we multiply this force by the transpose Jacobian, the desired torques at joint level are obtained. Since no torque loops exist, it is assumed that torque is proportional to the motor current, and therefore current references are generated for the motor drivers.
An important point is that the Jacobian used to compute the estimated dynamics parameters at task frame is suitably "extended" in order to take into account the additional constraint to solve arm redundancy (see [2]).This also means that the 7th degree of freedom is treated as the other ones (but always position-controlled) Arm redundancy constraint is computed according [3], and this solution seem the most appropriate for the arm to be used (see Fig. 4).

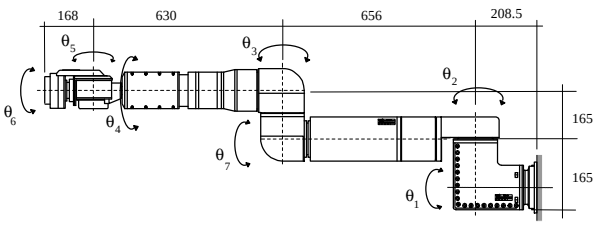

Fig. 4 Arm Architecture

This scheme for redundancy resolution has been proved to work well in simulations. An important point is that this scheme is inherently "drift-free", that means that close cartesian trajectories always correspond to close joint trajectories, differently from other common schemes for redundancy resolutions based on Jacobian transpose or pseudo-inverse. Of course this is very important in situations where a high degree of autonomy is required and where the arm posture shall be kept under strict control (and not only the cartesian position).
The resulting equations for the control scheme are:

$$F_{command} = \Lambda(q)\Omega F_m + m(x, \dot{x}) + \Lambda(q)\tilde{\Omega}F_s + \Lambda(q)\tilde{\Omega}F_a$$
$$F_m = -k_v(\dot{x} - \dot{x}_d) - k_p(x - x_d)$$
$$F_a = F_d + k_f(F_d - F_c)$$
$$F_s = -k_{v2}\dot{x}$$

$$\Gamma = J^T F_{command}$$

where a simple PD controller has been chosen for position and a P controller for the force. (see [4]). Ω is a 6x6 matrix which selects operational space directions where position is controlled. The counter part $\tilde{\Omega}$ selects directions of control of forces.

In G1 block, the sensors are read from the related boards via VME bus. A comparison between the sensor reading and the sensor estimation, provided by M1 block, is also done.
In M1 block all the robotic data acquisitions are managed. Data, written in the global memory at the highest possible frequency by M1, is read at the desired frequency ant recorded in suitable memory areas, together with their time tags. Once acquisition is completed, the supervisor block is notified and is able to dump, under operator request, the file to ground.

3.3 Emergency Controller

Emergency controller SW has the same structure as nominal controller with some simplifications. In particular, no cartesian loops exists, since the arm motion can be performed only one joint at a time (and the others switched off). This means that only commands for single joint motion exist. In detail:

- MOV_JOINT_P: moves the specified joint in position mode, to a specified setpoint
- MOV_JOINT_NC: moves the specified joint using the specified current and specified motor angular speed, for a given time

The H4 block reduces to reading these commands and pass them in binary to the E-Move level after a syntax checking.
Another important architectural difference is that the communication between the main CPU and the other boards are performed via RS485 bus, in a “one master-more slaves” fashion.

3.4 Microcontrollers for Safety

Microcontroller for safety are used to perform the required checks for the hazards control. Two microcontrollers are foreseen for the nominal mode, while only one for emergency mode. Checks to be performed are related to:

- Position: check that arm parts do not protrude the assigned envelope;
- Speed: check that the speed of each joint do not overcross given thresholds;
- Resolver cross check: check that the resolver readings for each joint (e.g. motor, coarse output shaft and fine output shaft resolvers) are coherent;
- Alive: check that the other boards are correctly running;
- Threshold: check that the safety thresholds, stored in 3 different locations, are not corrupted by Single Event Upset (SEU).

3.5 Safety and Contingency Aspects

Besides standard safety hazards (structural failures, contamination, etc.) which are applicable for an externally exposed payload, there are some specific hazards which are applicable to a robotic system (see [1]), in which the controller plays a major role:

- *Arm parts out of defined workspace*: EUROPA will provide a two-fail safe mechanism to avoid collision with surrounding objects and protrusion outside the assigned workspace;
- *Collision effects*: EUROPA will provide a two-fail safe mechanism to mitigate the effect of a collision to a maximum value for impact load and energy;
- *Impossible arm stowing*: to minimize the probability of Astronaut intervention in EVA, the EUROPA robot controller will be provided with redundant hardware (emergency controller) which intervenes when there is a failure on the nominal part (nominal controller) that cannot be recovered.

The control of the first two hazards (*Arm parts out of defined workspace* and *Collision Effects*) cannot be performed solely by hardware without jeopardizing the execution of the experiments (e.g. the use of mechanical hard-stops would seriously limit the manipulator workspace). For this reason, a computer based control system (composed by three independent computers running in parallel) will be used for hazard control.
According to NASA requirements, a computer system is considered zero-fault tolerant in controlling a hazardous system (i.e. a single failure will cause loss of control) unless independent computers are used, each executing uniquely developed functionality [6].

3.5.1 Arm parts out of defined workspace

To control this hazard, the following requirements have been defined at system level:

a. During non-contact motion, any movable part of EUROPA manipulator shall stay at a distance from the surrounding environment and from the ExPA envelope boundaries of not less than a minimum safe distance.
b. During contact motion, any movable part of EUROPA manipulator shall stay at a distance from the ExPA envelope boundaries of not less than a minimum safe distance.
c. During contact motion, the elbow limbs of EUROPA manipulator shall stay at a distance from the surrounding environment of not less than a minimum safe distance.

The limitation of the arm workspace is conceptually represented in Fig. 5.

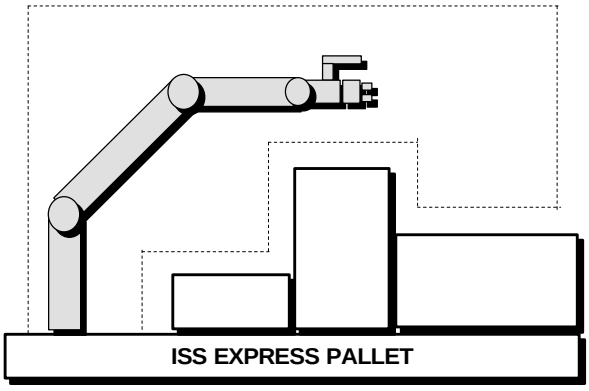

Fig. 5 Arm workspace limitation

The minimum safe distance has been preliminarily set to 0.02 m.
The minimum distance will be computed between a geometric model of the environment/ExPA envelope and a geometric model of the robot in the actual pose (obtained from motor/joint position sensors and forward kinematics). The position error hazard cause will be controlled by implementing a two-fail-safe computer based control system.
The two-fail safe system will be achieved as described in Fig. 6.

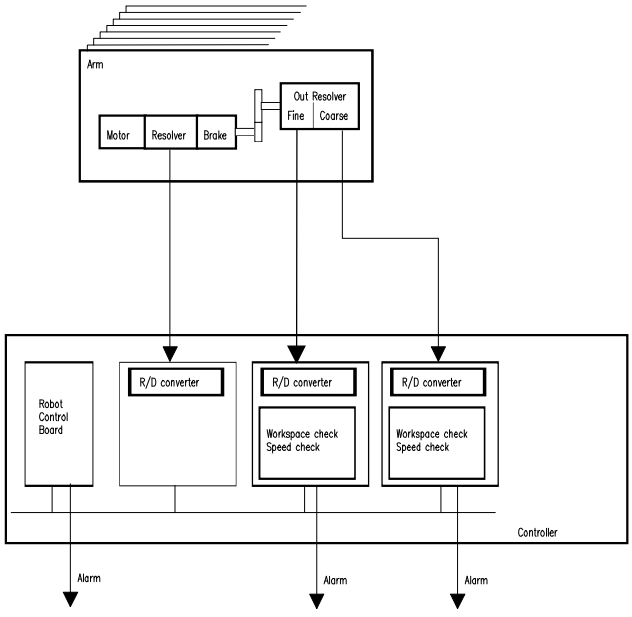

Fig. 6 Two-fail safe control system for position and speed

The arm provides three independent position sensors: a motor shaft resolver, an output shaft coarse resolver and an output shaft fine resolver. The three independent sensors are acquired by three independent computer boards:

- The Robot Control Board (MPC750 processor based board, through the VME bus and the motor driver boards);
- The two microcontroller for safety boards.

Each of these boards will run a specific software to compute the minimum distance between the arm and the environment, based on the following algorithm:

- Compute the position of each link of the arm, starting from the joint position sensor (resolver) through forward kinematics;
- Compute the minimum distance between each link of the arm and each environment object to be checked for collision;
- Check that the minimum distance is above a given threshold. In case it is below the threshold, generate a stop command.

Two different thresholds can be implemented, a warning and an emergency threshold, respectively leading to a controlled stop and an emergency stop of the arm. Controlled stop is defined as stopping the arm in a controlled manner by applying a smooth deceleration profile and while remaining on the desired path. Emergency stop is defined as stopping the arm by applying all brakes and disabling motor drivers.

The emergency stop threshold will be computed by adding to the minimum safe distance the arm emergency stopping distance, the arm travelling distance due to discrete time implementation of the check and other possible margins due to imperfect resolver calibration, etc.
The controlled stop threshold will be computed by adding to the emergency stop threshold the arm controlled stopping distance plus a small margin to ensure that the emergency threshold is not violated when performing a controlled stop, thus avoiding the violation of the emergency threshold and the abrupt emergency stop of the arm.

3.5.2 Collision effects

To control this hazard, the following requirements have been preliminarily defined at system level:

a. EUROPA robot arm shall not cause impacts between any part of itself and the surrounding environment having an energy in excess of 0.1 J.
b. The EUROPA robot arm shall not cause impact loads greater than 250 N between any part of itself and the surrounding environment.

The impact load and energy will be controlled by implementing a two-fail-safe computer based control system against joint speed error and motor current error hazard cause. The joint speed and motor current will be kept below predefined thresholds.

The two-fail safe speed control system will be achieved as described in Fig. 6, by using again the three independent position sensors (resolvers) provided by the robot arm and the three independent computer boards (the robot control board and the two microcontroller for safety boards.
Each of these boards will run a specific software, implementing the following algorithm:

- Compute the speed of each joint, by numerical derivation of the joint position sensor;
- Check that the actual speed is below a given threshold. In case it is above the threshold, generate an emergency stop command.

The two-fail safe motor current control system will be achieved in the following way:

- Two independent hardware devices on each motor driver board will check that the current is always below a given threshold
- The Robot Control Board will contain a current saturation function (so that it will never generate a current setpoint higher than the threshold) and will acquire the motor current from the driver boards and check that it is below the threshold.

3.5.3 Impossible arm stowing

To control this hazard, the following requirement has been defined at system level which is applicable to the EUROPA robot controller:

EUROPA design shall be one-failure tolerant for the automatic execution (i.e. without requiring EVA intervention) of the Stow function, except for the following cases:

- structural failure (when a function is implemented using structure, the structure shall be exempted for failure tolerance requirements);
- failure on robotic arm/end effector motor, brake, gearbox or bearings;
- other specific failures like jamming, ISS interfaces, etc.

Failure of one of the above items could require astronaut intervention in EVA.
Failure tolerance is here focused toward minimization of astronaut intervention in EVA. Apart from the specific failures listed above, EUROPA shall provide two functional chains for the execution of the Stow function, a nominal controller and an emergency controller, corresponding to nominal control mode and emergency control mode of operation. In emergency control mode, it will be possible to move the arm one joint at a time using redounded capabilities of the motor drivers. The nominal controller and the emergency controller functions have been described in paragraphs 3.2 and 3.3.

3.5.4 Changing safety thresholds

In the description of the computer based control system used for hazard control, some basic safety parameters have been identified, referred to as safety thresholds. They are motor current thresholds, joint speed thresholds, arm-environment minimum distance thresholds and workspace definition.
During the execution of the on-board experiments with the EUROPA manipulator, there might be a need to make use of different values of the safety thresholds.
For instance, during non-contact motion, higher speed limits may be accepted, provided that sufficiently low motor current limits are used (the impact load and energy depends on the combination of current and speed limits). During contact motion, instead, higher current values might be required to perform robotic manipulation tasks, while the speed can be kept to very low levels.
As far as the workspace limitation is concerned, EUROPA must remain within an envelope of 864 x 1168 x 1245 mm). Protrusions outside the envelope could cause collisions with adjacent ExP payloads or field-of-view obstructions. Any exceedances to the ExP payload envelope make a payload non-standard and will have to be approved by the ISS Program. Payload envelope exceedances will be pre-planned operations, and neighbouring ExP payloads will be informed about any permanent or intermittent payload envelope exceedances. Therefore, it is necessary to guarantee that EUROPA will perform all of its operations inside the assigned envelope (standard or non-standard). On the other hand, if there is a need to temporarily protrude from static envelope, due to particular task requirements or due to the need of operating adjacent payloads, it will be possible for EUROPA to modify on-orbit the workspace definition. For instance, if a top protrusion of 0.1 meters is needed, the workspace definition can be accordingly enlarged.

The possibility of changing safety thresholds in general requires the involvement of the ground operator. It will be necessary to implement a safe transmission protocol which allows to be sure that, when a safety parameters has to be updated, it will be updated in all the three on-board computers composing the EUROPA computer based control system in charge of safety checks.

The main goal of such a protocol can be summarized in the following way: when there is a transmission chain of computers in charge of transmitting a given parameter

between a terminal computer from one side (e.g. the operator interface on-ground) and another terminal computer from the other side (e.g. a processor board in charge of safety checks), no failure or software error occurred at any step of the transmission by anyone of the involved computers can be not detected by the terminal computers (hence all possible errors / failures will be detected). The concept of the safe transmission protocol is shown in Fig. 7.

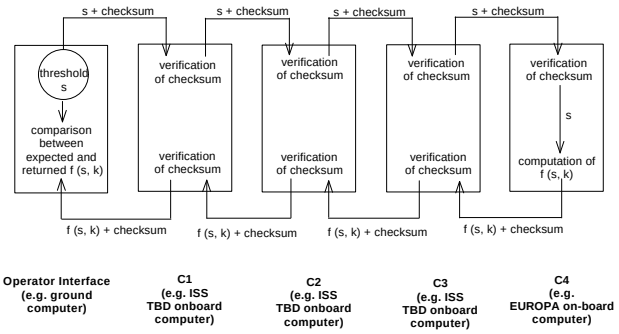

Fig. 7 Safe transmission protocol

It basically consists of the following steps:

- the operator interface transmits the parameter *s* to the destination computer (through the transmission network);
- the destination computer returns a transformed value *f(s,k)*, function of the parameter *s* and of a specific key *k*, only known by the operator interface and the destination computer;
- the operator interface receives the transformed value and compares it with the expected value, obtained by locally applying the same transformation algorithm;
- at the next step, the value of *k* is updated, using a law only known by the operator interface and the destination computer.

This algorithm allows to avoid possible single points of failure due to the fact that the transmission chain unique (not redounded) and a failure on one of the computers in the chain could lead to safety hazard, if not properly controlled by this protocol.

To give some examples, the protocol is robust against the following classes of errors:

- transmission of wrong data;
- inadvertent activation of an update command performed by an intermediate computer in the chain (for instance by sending twice a command due to software error or program stack pointer corruption by ionizing radiation effects);
- masking of telemetry data by an intermediate computer.

4. Conclusion

The EUROPA robot controller will provide a hardware and software platform, fully dedicated to on-board control of the EUROPA robot arm, based on state-of-the-art robotic technologies. It will allow to implement advanced control algorithms and reach a high degree of autonomy for the on-board operations.
In addition, it will make use of a computer based control system fail safe approach, which provides all the necessary features to ensure the safety of the robotic operations while it still preserves their flexibility and does not prevent manipulator dexterity (workspace limits are not fixed by hard stops or mechanical limit switches, but can be changed depending on the application, allowing to potentially operate on other ISS payloads).

References

[1] A. Rusconi, R. Finotello, T. Grasso, R. Mugnuolo, A. Olivieri, “EUROPA Robot Controller”, in *Proceedings of i-SAIRAS 2001* (St-Hubert, Canada), June 18-22, 2001

[2] J. Baillieul, “Kinematic programming alternatives for redundant manipulators", in *Proceeding of.IEEE Int. Conf. Robotics Automat.*, St. Louis, Mar. 1985, pp. 722-728

[3] H. Seraji, “Configuration Control of Redundant Manipulators: Theory and Implementation”, in *IEEE Trans. on Robotics and Automat.*, vol. 5, no. 4, August 1989, pp. 472-490

[4] O.Khatib “A unified approach for motion and force control of robot manipulators: The operational space formulation”- , *IEEE Journal of Robotics&Automation* vol. RA-3, No. 1, February 1987

[5] A. Rusconi, R. Finotello, G. Borghi, R. Mugnuolo, A. Olivieri, F. Pasquali, "EUROPA (External Use of Robotics for Payloads Automation)", in *ISS Utilization Conference - Research Using the EXPRESS Pallet - Part 1* (Cape Canaveral, Florida) October 15-18, 2001

[6] “Computer Control of Payload Hazards”, MA2-97-083, Space Shuttle Program Integration, 1997.