Evaluating a Flexible Architecture for Distributed Control

P. Bellini*, M. Buonopane*, M. Montanelli#, P. Nesi*
*Department of Systems and Informatics, University of Florence, Italy
Tel. +39-055-4796523, Fax. +39-055-4796363, nesi@ingfi1.ing.unifi.it, nesi@dsi.unifi.it
#SED, Special Electronic Design, email: c.bruni@sednet.com
Project www site: http://www.dsi.unifi.it/~hpcn/wwwmupaac/wwwpag.html

Abstract

Along production pipelines in manufactory industries several elaboration phases in which the movement of interpolated axes are present. These activities have to be synchronized with several other simpler activities along the pipeline. Computerized numerical controls for such systems have to be flexible and strongly expandable and reusable since pipelines are frequently reconfigured to realize differently arranged pipelines with different technical requirements. In this paper, the assessment of such a flexible architecture is presented on the basis of which the building of flexible distributed controls for pipelines is possible. The architecture has been defined in the ESPRIT HPCN (High Performance Computer Networking) project MUPAAC (Multi Processor Architecture for Automatic Control). MUPAAC architecture and prototype has been assessed in order to identify the most important performance indexes.

Keywords: numerical control, distributed control, performance evaluation, systems assessment, CNC, PCI, communication, CANBUS, fieldbus.

1. Introduction

Presently, most of the automatic machines (for milling, cutting, measuring, soldering, electron erosion, moving pieces, robots, tapes, etc.) used in pipelines of production are controlled by microprocessor-based systems. These are usually called CNCs (Computerized Numerical Controls) and provide a certain number of interpolated axes and digital sensors/actuators [1]. The CNCs are typical real-time systems [7], [8], with all their classical problems.

In general, a CNC system receives the instructions describing the elaboration to be performed in terms of an ISO 1037 program. The ISO programs are generated by CAD/CAM stations. The ISO program is composed of elementary instructions that include instructions to specify the *macropoints* for interpolation (plus eventually some technical information). The CNC system, by means of an interpolator, calculates the *micropoints* that are useful to generate the profile required between the given *macropoints*. The operation of a CNC can be summarized as follows:

- Interpretation of ISO programs and interpolation. Specific versions of ISO programs can be used for describing both the operation made by the machine and by its I/O ports in terms of logic equations;
- Execution of low-level commands for axes of the machines and for the other auxiliary services by means of several digital and analog ports (sensors);

The CNC is capable of (i) detecting errors/faults that may occur on the machine, and (ii) monitoring its production. Errors are corrected by a special set of instructions and/or are reported towards the other microprocessor-based systems that are monitoring the automated area.

CNCs are used as the elementary component for building complex production pipelines. These can be organized in three hierarchical levels.

The first is usually defined as FMM (Flexible Manufacturing Module) -- i.e., the production islands. FMMs are constituted by a single machine tool with auxiliary robots for charging and discharging pieces and tools. When the required performance is low, the FMM control may consist of a single microprocessor-based system that coordinates the machines and the auxiliary tools and services (e.g., 3 interpolated axes for the machine plus 2-3 axes for each robot). When the provided performance is high, the FMM has to be managed by a multiprocessor system. This solution highly increases the complexity of the system for both hardware and software aspects. In these cases, the production process of each FMM is managed by a central control station. All the FMMs of the production pipeline have their own controllers that talk with a centralized system of control and co-ordination by means of a local communication network.

The second level is the FMS (Flexible Manufacturing System). It is made by several FMMs connected by a communication channel to a central station that manages the various production cells and the production flow. Among the most important CNC producers, SIEMENS proposed a network based on IEEE 802.3. Other firms

such as OMRON, ABB, Blue Chip have their own communication networks for connecting only PLC (Programmable Logic Controller) modules. Intelligent Instrumentation Inc. uses an Ethernet network to connect its own *field bus* modules such as Profibus, Bitbus or CANBUS (Control Area Network BUS). These connections are master-slave. The master manages the communications, thus a typical centralized non-distributed controller is obtained.

At the third level, the Factory General Supervisor (FGS) is present. Frequently, the role of FGS is covered by a couple of specialized machines one for managing the administrative part and the other for controlling the production process.

Please note that, the strong integration among layers of control and the different machine tools is frequently claimed by CNC producers. In effect, it is an illusion, since it is possible only by using modules with homogeneous performance. Most of the production pipelines are specifically set up for a given production, and then they are destroyed when the production is concluded, in favor of a new pipeline.

1.1. Context

Presently the complexity of FMSs is growing since a higher number of robots and services are needed and these cannot be confined in FMMs. Some of these services need many axes for their control. This local complexity of control cannot be managed by using decentralized numerical controllers since the remote connection by means of serial or specific channels is not fast enough for coordinating axes. For these reasons, some builders are improving the capabilities of their CNCs by increasing the power of the microprocessor.

The general diffuse problems of the above mentioned solutions is the lack of flexibility (see also OSACA and NETCIM EC projects). At the first level, the number of controlled axes and their performance can change radically from a machine to another. The solution to provide a controller per axis is too expensive and inflexible to be acceptable. At the second level, the controller of FMM can be in some cases even unuseful and under-exploited. While, the communication between the FMM controller and the other CNCs has to be provided by means of high performance communication channels and not only by Field Busses, which have a limited bandwidth.

More recently, few builders of CNCs in USA are beginning to study multiprocessor-based systems (parallel architectures) for implementing flexible CNCs and FMSs. This flexibility also reduces the complexity and the cost/price of complex numerical controls and increases the modularity of the system.

According to the above mentioned CNC builders, s*olutions based on high performance CNC cannot be re-used for controlling more axes in slower pipelines of production since they are inflexible; solutions based on low performance CNCs cannot be composed for controlling high performance machines in high performance production pipelines. Systems including both low and high performance CNCs cannot be built at reasonable costs since the restructuring of a pipeline of production frequently provoke the inclusion of new machines with different CNCs that cannot be easily integrated each others.*

Moreover, these systems present several variables that influence their performance on the basis of the configuration chosen. This approach can be inverted to look for a configuration that have the required performance at the lowest price. Thus, the evaluation of these critical systems has to be carefully performed. A detailed evaluation and model allows identifying the correct configurations and the final performance.

In this paper, MUPAAC (Multi Processor Architecture for Automatic Control) architecture is presented together with its assessment. MUPAAC HPCN ESPRIT IV project has been developed in order to solve most of the above discussed problems [2], [3]. HPCN technology has been employed in defining the parallel and distributed architecture MUPAAC. A solution has been found by studying and implementing a set of more flexible components that can be reused in a variety of combinations for covering from low to high performance. The evaluation reported has been performed for validating the results obtained by the project, which has been successfully concluded in February 1999. MUPAAC is suitable for implementing CIM (Computer Integrated Manufacturing) solutions. The discussion of CIM policies is not on the focus of this paper.

The MUPAAC solution is a three layer architecture. A FGS controls the whole production pipeline by means of a set of industrial computers connected via local area network. Each industrial computer can control one or more machines by means of a set of specific DSP-based boards. These specific boards and the industrial computers control all digital inputs/outputs, sensors and actuators of the pipeline of production via CANBUS. It allows the construction of a variety of configurations.

The partners of MUPAAC project have been: SED Inc.; University of Florence with the Departments of Systems and Informatics and that of Electronic Engineering; VALIANI Inc. (as end-user and validator); and CESVIT (High-Tech Agency) as TETRApc-TTN HPCN (Technology Transfer Node) partner. The prototype produced has been tested and validated by VALIANI in the specific field of builders for cutting machines for producing passpartout (e.g., ZUND, Swiss; Gunnar, Swiss).

The paper is organized as follows. Section 2 presents the general MUPAAC architecture mainly considering

hardware aspects. Section 3 presents the software architecture of MUPAAC. Section 4 reports the System Evaluation as Performance Analysis of the most relevant parts of MUPAAC architecture. Conclusions are drawn in Section 5.

2. General Architecture

MUPAAC architecture presents a further level with respect to the architectures proposed by other CNC builders. A general machine, (MUPAAC Supervisor), controls the whole pipeline of production by means of a set of Special Industrial Peripheral Computers (SIPCs) connected via local area network (see Fig.1). Each industrial computer can control one or more machines (for milling, cutting, measuring, soldering, electron erosion, moving pieces, robots, tapes, etc.) by means of a set of DSP-based boards. These and the SIPCs control all digital input/outputs, sensors and actuators of the pipeline of production via CANBUS. A high flexibility is reached by allowing the construction of a distributed control by using a variety of configurations. From the single controller to a set of 256 industrial computers each of which may control up to 4 industrial machines with at most 4 axes each (16 axes per SIPC thus up to 4000 axes). The distributed control can be reconfigured in order to satisfy changes of configurations in the production pipeline.

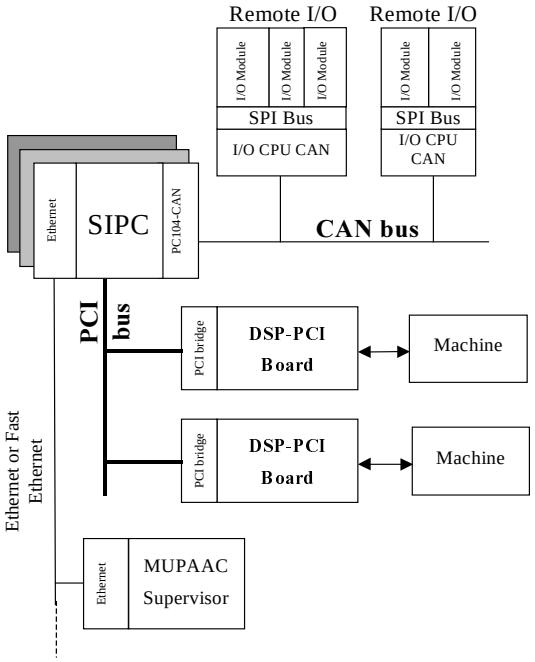

Figure 1. Hardware architecture of MUPAAC

In MUPAAC, the flexibility has been reached by the:

(i) structure of hardware and software of DSP-based boards that allow to manage from 1 to 4 axes;
(ii) management of inputs/outputs from both the industrial computer (controller of the FMM) and the DSP-boards;
(iii) reconfigurability of software for managing the mapping of axes and inputs/outputs;
(iv) adoption of fast connections between the FMM controller and the DSP-based boards and the FGS and the FMMs via Fast Ethernet;
(v) *plug and play* solutions for PCI and CANBUS components.

The whole-distributed control can be reconfigured in order to satisfy changes in the production pipeline configuration with fully reusing the same boards for covering any kind of performance needed. MUPAAC architecture includes both hardware and software aspects. The hardware architecture is shown in Fig.1, where the main components are reported.

MUPAAC-Supervisor is a general purpose workstation, the user interface of the entire system and the connection with CAD/CAM area. It sends/receives messages to/from the actual microprocessor based systems for controlling the machines via SIPC.

SIPC boards (Special Industrial Peripheral Computer) are microprocessor-based systems that execute ISO instructions coming from the MUPAAC Supervisor. SIPC also interacts with (i) the DSP-PCI boards for controlling axes and receiving alarms and synchronizations; (ii) the Remote I/O boards for activating and receiving I/O signals via CANBUS. The CANBUS board is based on PC104 (ISA like bus) interface while the Network Card can be either PCI or ISA.

DSP-PCI boards are based on the Analog Device AD2106x (SHARC) DSP, for managing up to 4 axes of motors. At the physical level, the communication between SIPC and its DSP-PCI boards is made via PCI bus. The boards support *plug and play* configuration mechanism, so boards can be plugged directly in without any manual or software configuration (DIP-Switches, etc.) to avoid conflicts (interrupt line, memory map).

Remote I/O boards allow the reading/writing of I/O ports. They are endowed with a microprocessor for interpreting messages sent on CANBUS, which are specific commands for managing I/O ports and managing the *plug and play* mechanism. On the CANBUS up to 64 Remote Is/Os can be attached. Each Remote I/O board can have on its CAN-interface and up to 8 I/O Modules via SPI bus (Serial Peripheral Interface). Therefore, a SIPC can indirectly have up to 512 I/O Modules. Two different CAN-interfaces have been realized: (1) with an Intel 8051 for I/O modules with a limited needs of calculation power, and (2) with Hitachi SH7000 for satisfy high demand of calculation. Different types of I/O Modules are possible for digital inputs, digital outputs, outputs with relays, counters, encoder, analog inputs, analog input, and low performance monoaxis control. These boards support a kind of *plug and play* configuration mechanism, since the I/O modules have

special attributes that permit to know the types of the modules.

CANBUS is a 1Mb/s serial communication bus for establishing communications between each SIPC and its I/O boards.

PCI BUS is a specific communication support based on PCI. It is used by each SIPC board for communicating with its DSP-PCI boards.

TCP/IP-based Network is an Ethernet or Fast Ethernet network based on TCP/IP.

MUPAAC solution has been obtained by reengineering both hardware and software of INDEX-DSP architecture of SED. This used only one DSP to control up to 4 interpolated axes and an i486 CPU to control the whole system. For the reengineering of the software components the classical techniques for monitoring and planning the activities have been adopted [1], [10], [11]. A preliminary evaluation of potential worst performance that could be reached by the new architecture was performed. The actual performance obtained had been better than those supposed in several configurations.

2.1. Flexible Configuration

In Tab.1 a set of possible configurations of MUPAAC architecture are presented. The configurations present from 4 to 1024 axes.

- NA: Number of Axes of the whole system;
- ND: number of DSP-PCI boards in the whole system;
- NAD: Number of Axes per DSP-PCI board (at most 4);
- NS: Number of SIPC boards with NDS DSP-PCI boards each;
- NDS: Number of DSP-PCI boards for each SIPC board (ND/NS) (at most 4);
- C: Cost factor evaluated considering C = 2 NS + NDS (the cost of the SIPC has been considered double with respect to that of DSP-PCI board).

Table 1 - Example of MUPAAC configurations

NA	ND	NAD	NS	NDS	C
4	2	2	1	2	4
4	1	4	1	1	2
4	1	4	1	1	2
4	1	4	1	1	2
4	1	4	1	1	2
8	4	2	1	4	6
8	2	4	1	2	4
16	8	2	2	4	12
16	4	4	1	4	6
32	16	2	4	4	24
32	8	4	2	4	12
64	32	2	8	4	48
64	16	4	4	4	24
128	32	4	8	4	48
256	64	4	16	4	96
1024	256	4	64	4	384

MUPAAC architecture is flexible since can fully be reconfigured for satisfying the requirements of the evolution of production pipelines. It can be suitable for covering both low and high performance control networks. Moreover, axes, DSP-PCI boards, SIPC boards can be added/removed when needed/unuseful.

The configurations reported in Tab.1 do not take into account the possibility to have different performance on different SIPC and DSP-PCI boards. Different performances are obtained by using a different number of axes for each DSP-PCI board, and by using a variable number of DSP-PCI boards per SIPC. Thus, they are purely indicative of the actual flexibility of MUPAAC.

3. Software Architecture

The software architecture of MUPAAC system is reported in Fig.2. The several components are distributed on the hardware elements according to the general architecture.

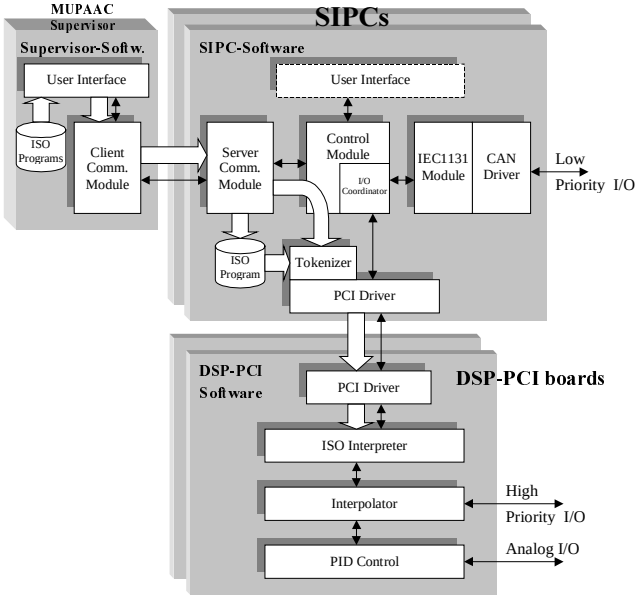

Figure 2. Software architecture of MUPAAC

3.1. Supervisor Software

The Supervisor Software presents a simple *User Interface* that enables the users to (i) send ISO programs and single ISO commands to each specific SIPC, and (ii) to view errors/alarms collected by the SIPCs. The *Client Communication Module* implements the client part of a socket-based custom communication protocol. This permits to send ISO Programs (as text files) or single ISO Commands (as strings) to SIPCs. It permits to receive errors, alarms and the end-of-processing notification. This part has been implemented under Windows NT but can be easily ported to UNIX or any other platform. According to our experiments, this part is not critical for the system performance, even for a high number of SIPCs.

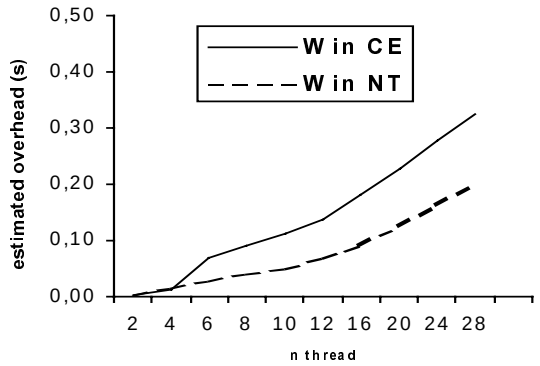

Figure 3. Estimated overhead of WinCE and Windows NT

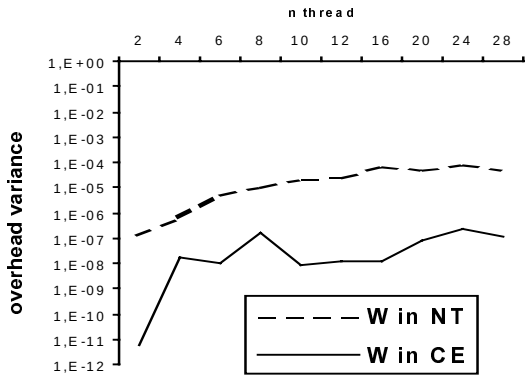

Figure 4. Overhead variance of WinCE and Windows NT

3.2. SIPC-Software and SIPC-PCI Protocol

The SIPC is endowed with Microsoft WindowsCE 2.1. It is quite suitable for embedded systems since it is can be customizable and ROMable, and presents TCP/IP. It is also a multitasking preemptive operating system quite suitable for soft real-time applications. Before to adopt WindowsCE, its performance and reliability have been assessed by using methodology and measures proposed in [4], [5], [6].

In Fig.3 and 4 the overhead and the variance of the overhead introduced by the scheduler while scheduling threads of the same process, is reported. In Fig. 3 and 4 WindowsCE has been compared with Windows NT 4.0. The overhead of Windows NT is lower than that of WindowsCE, but the variance and so the predictability of the scheduling is much better for WindowsCE than that of Windows NT. This is due to the several uncontrollable tasks that are typically executed on Windows NT at kernel level. In [6], Windows NT has been compared with LINUX, OS/2 and Win95.

The *User Interface* for the SIPC is optional, since the operations can be remotely done from the Supervisor PC. It was developed only for debugging purposes, but a local emergency/supervising interface may be useful.

The SIPC-Software component is mainly used to:

1. communicate with the Supervisor to receive the ISO Program to be executed;
2. notify errors and alarms to the Supervisor;
3. coordinate the I/O management;

These activities have been assigned to independent thread of execution. The first two present the same priority while the third is temporized to start every a predefined number of milliseconds.

The *Server Communication Module* implements the counterpart of the protocol used to receive the ISO Programs/Commands. The ISO program is temporally stored in a RAM file and then each instruction of the program is sent to the proper DSP-PCI board. The program is stored locally since the same program has to be frequently repeated for the production.

Before sending an ISO command/entity to the DSP-PCI board, it is tokenized (by the *Tokenizer*). The string representing the command is coded with a sequence of numbers for reducing its dimensions.

The ISO commands are sent to a DSP-PCI board through the PCI bus by using a specific SIPC-DSP Protocol. The communication with the DSP is interrupt driven and is based on a dual port RAM for data communication. The dual port memory is subdivided in three regions, one used to transmit data from the SIPC to the DSP, one to receive, and the last presents the status region. In this region, (i) the values of inputs and of the outputs of the CAN, (ii) the flags associated with the axes and (iii) the ideal and real quota of the axes, are stored. When the SIPC writes a message, it generates the interrupt to the DSP and waits for an ACK from the DSP. This protocol has been implemented in a specific WindowsCE PCI Driver for the DSP-PCI board. Another important task of this module is to send error and alarms messages to the Supervisor that have been received from the DSP-boards, the Remote I/Os or internally generated.

Another task of this component is used to coordinate the *DSP-boards* with the *Remote I/O Modules* connected with the CANBUS (*I/O Coordinator Module*). This module has to update regularly (every 10ms) the outputs and to get the inputs; it uses the *CAN Driver* to access to the *Remote I/O*. This has been implemented with a specific WindowsCE driver for the PC104-CAN board. The outputs can be set from the DSP-boards (by updating the Status region) via the logic equation solver (*IEC1131 Module).* The logic equation solver gets the inputs values, the flags, the real and ideal quotas of the axes (using the *PCI Driver* to access to the status region of DSP-PCI board) and produces the outputs to be sent to the Output Modules on the CAN based on particular logic conditions.

3.3. CANBUS Protocol and Remote I/O Boards Software

The CANBUS is a serial communications protocol that efficiently supports distributed real-time control with a very high level of security. Its application domain ranges from high-speed networks to low cost multiplex wiring. In automotive electronics, engine control units, sensors, etc., are connected using CANBUS with bit-rates up to 1Mbit/s. A CANBUS node has a layered structure:

Application Layer
Object Layer
Transfer Layer
Physical Layer

The goal of the *Physical Layer* is the transfer of the bits between the different nodes with respect to all electrical properties. Within one network physical layer, of course, it has to be the same for all nodes. The *Object Layer* and the *Transfer Layer* comprise all services and functions of the data link layer defined by the ISO/OSI model. The *Application Layer* is used to establish communication between applications. This level has not been standardized; and various protocols have been presented (SDS, CanOpen, DeviceNet).

In MUPAAC, the Intel 82527 chip was chosen for the CANBUS module for realizing the Physical, Transfer and Object Layers. For the Application Layer was chosen to implement a SDS subset, called MUPAAC-SDS.

The communication model supported by MUPAAC-SDS is the master/slaves communication, where the master device (the SIPC) uses the I/O services provided by the slave devices (the Remote I/O). The master views each slave device as an object with:

- a set of attributes that may be read or written;
- a set of actions that may be called;

A slave device is identified by: Device Address (6bit); and the EOID (Embedded Object Identifier) (4bit);

In MUPAAC-SDS are implemented three Application Layer services:

- The **Read Service** to read an attribute value of an Embedded object. For example, this service could be used to read the sensor value.
- The **Write Service** to modify an attribute of an Embedded object. For example, this service could be used to set an actuator output.
- The **Action Service** to execute the actions specified for an Embedded object. For example, this service could be used to move an axis to a target position.

Data transmission can be Basic (for data with length less or equal to 6 bytes) or Fragmented (for data greater than 6 bytes). When a Request (Read/Write/Action) is sent from the master (SIPC) a Result (positive or negative) is sent from the slave (Remote I/O).

The master of MUPAAC-SDS protocol has been directly implemented in the *CAN Driver* on Windows CE. It exports to the application the functions needed to Read/Write attributes and to execute actions on remote I/O ports. Specific software has been implemented for i8051 and SH7000 CPU located in the Remote I/O CAN-interface boards. Mainly, it has to interpret the requests received from the SIPC (read an attribute, write an attribute, etc.) and to send a response or an ACK.

In the case of a complex configuration, a part of the workload of the *IEC1131 Module* of the SIPC may be demanded to the SH7000 CPU on to a CAN-interface board. In fact, it may be programmed to resolve autonomously some logic constraints among its input and output modules.

3.4. DSP-PCI Software

The DSP-Software component is used to receive the tokenized ISO Commands from the PCI bus (trough the *PCI Driver*) and to manage their execution (see Fig.5).

Two types of interpolations can be used on DSP software, the linear and the circular one.

The commands received from the SIPC are stored in the *Raw Command Queue* and then are interpreted by the *Macro Instruction Interpreter*. The processed instructions with additional information are then stored in the *Interpreted Command Queue,* ready to be used by the interpolator.

If the received command is for moving axes, it is firstly elaborated for calculating some important data for the interpolation. For example: the length of the trajectory to realize, the test for the continuity between two movements, calculus of the movement velocity, center coordinates of the circle (only for circular interpolation), etc. In the case of a non-movement command, they have to be checked if must be executed immediately or if must be executed after a complete stop of the axes. In the first case, the command is executed immediately, else the end of the previous movement has to be waited, and then the command requested can be performed. Example of a command executed after a stop of the axes is the change of PID (Proportional Integrative Derivative control algorithm) parameters. In the opposite, an example of command that may be executed before the stop of the axes is the transmission on the PCI bus of the value read from a space transducer (encoder). The DSP-PCI board software also controls the analog outputs of the board which are connected to the motor's power-driver.

On the DSP-PCI board, some inputs and outputs must be directly controlled by the interpolator software for their importance for the machine safety. They are the enabling bits of the motors, the limit switches, encoders zero signals. These I/O signals with the Servo Error (maximum trajectory error allowed by the system) are controlled every servo cycle (bottom part of Fig.5) before the

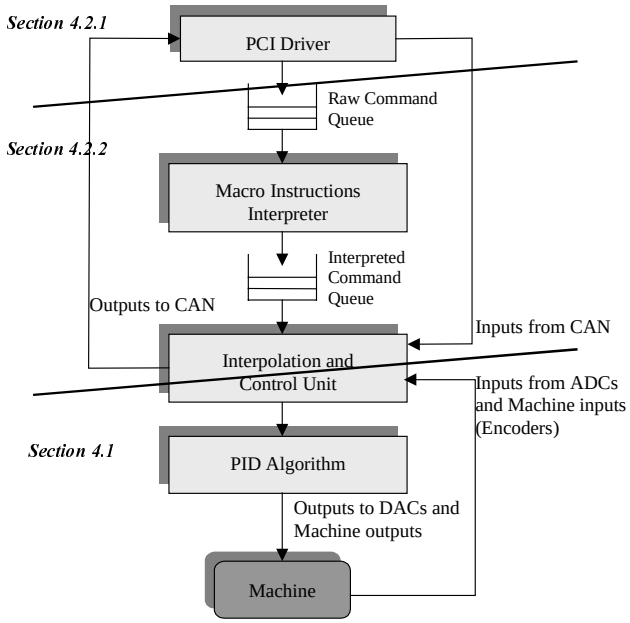

Figure 5. DSP-PCI-Software modules (execution costs for each part are discussed in the corresponding section)

micropoints generation.

The interpolator must also know other inputs from CANBUS. To this end, the DSP software may access the State region to get the last value assumed from an input that the interpolator want to know.

In the Interpolation phase, the micropoints that are used in driving the machine's motors to realize the piece profile are generated.

The algorithms used in this elaboration are critical for the quality of the system. In this phase, algorithms that guarantee the maximum precision rate in the elaboration must be used, and these algorithms must be as fast as possible. The velocity of the elaboration affects the precision of the elaboration.

The software realized for this board use EXACT DDA algorithm for the linear interpolation, while for the circular interpolation an interpolation with Sin Cosine has been adopted, because it guarantees to us more precision than the DDA one. In certain cases, more sophisticated interpolation algorithms can be used -- Nurbs, Splines.

The interpolation produces the micropoints that are the set point used by the PID algorithm. This is quite sensible to the elaboration time.

The above described operations are performed by three concurrent tasks: PCI driver, interpreting and interpolation, PID.

4. System Evaluation as Performance Analysis

CNC can be classified on the basis of the refresh time (period) for evaluating/generating the actions on axes, RefAx (Refresh Axes Time, servo cycle time). For lower values of RefAx higher performance and costs are provided, since a short refresh time leads to reduce errors (in cutting, soldering, etc., i.e., in following the planned profile) and executing elaborations in shorter time.

- **HIGH performance (RefAx < 1ms):** PMAC (USA), GALIL (Israel), Allen Bradley (USA), Siemens (Germany), NUM (France), etc.
- **MEDIUM performance (1 ms < RefAx <7 ms):** Fagor (Spain), Siemens (Germany), etc.
- **LOW performance (RefAx > 7 ms):** SIPRO (Italy), SED with INDEX-DSP architecture (Italy), ECS (Italy), Siemens (Germany), etc.

Please note that some CNC builders present several types of solutions that are capable of covering different sectors of the market -- e.g., Siemens, ECS, etc. Presently the controllers that are employed on cutting, soldering, moving, machine tools belong typically to MEDIUM and LOW categories. The adoption of high performance solution increases the precision of elaboration.

In order to show the main results achieved with MUPAAC prototype, some relevant parameters for the evaluation of motion controller systems have to be considered. These parameters are:

- Time of servo cycle axis (RefAx mentioned in the introduction).
- Number of Entities (elements of the ISO program) Processed per Second (**NEPS**) passed from the SIPC to the DSP board.
- Time of cycle for updating I/O via CANBUS.
- Flexibility of composition and reuse.

The first three factors are used to measure the goodness of a control system for production pipeline in terms of working precision. These metrics are typically considered by pipeline builders to identify the most suitable and powerful solutions.

The last aspect to be considered is the flexibility. This can be measured based on the range of applicability of the same component (as well as numerical control) with respect to the different performance categories. In project MUPAAC, a special attention has been given to solution flexibility and reusability. In these cases, if a CNC has flexible components it can be easily reused during the reengineering and the reconfiguration of production pipelines.

4.1. Servo loop cycle time

The software located on the DSP-PCI board is the only involved in the management of the axes control and thus of RefAx factor. By using external probes, it has been possible to measure the time elapsed between the beginning and the end of every control cycle. The DSP-PCI board can manage different types of interpolations, thus different execution times have been registered for

different interpolation algorithms. A shorted RefAx allows to increase precision.

The initial value obtained by the servo cycle time was 57μs for a circular interpolation of two axes, and 91μs adding other two axes in linear interpolation. The estimations were performed by considering a classical PID algorithm for motor controls.

The servo loop time for only one axis in linear interpolation is of 32μs. These values resulted better than those predicted in the early phases of the project.

In the following table, the detailed values of time for each single operation of the algorithm are reported.

DSP-PCI board main tasks	Time (μs)
Control Algorithm and I/O operations**	8 + 14 (per axis)
Acceleration Slope calculation time	7
Linear Interpolation	3 per axis
Circular Interpolation	7 per axis

** The I/O operations consider the DACs and the encoders (see Fig.5).

An improvement has been obtained by modifying the programmable logic chip for managing the Digital Analog Converter, DAC, of the velocity feedback. After this change, the servo loop cycle time passed from 57μs (as 8 + 28 + 7 + 14) to 53μs for a circular interpolation of two axes (49 for linear) and from 91μs to 83μs for 4 axes: two in circular and two in linear interpolation.

The number of axes controlled by a single DSP-PCI board can be from 1 to 4 obtaining performance from 32 to 99μs (4 axes in circular interpolation). On the other hand, RefAx has to be fixed for guaranteeing the exact controllability of the motors via the control equation algorithm. Thus, it can be fixed to 100μs by considering the worst condition as that with: 2 axes in circular interpolation plus 2 axes in linear interpolation, that is, 83μs. In this case, 17μs up to 100μs can be used by the other tasks to perform the interpretation and the communication via PCI. This means that, for each instruction (interpretation and communication) more than one RefAx cycle has to be performed.

With this performance the MUPAAC CNC can be located in the high performance category of numerical controls.

4.2. Number of Entities Processed per Second

The measure of NEPS shows the ability of the system to process geometric entities. The entities correspond to the elementary ISO instruction of the part program that the numerical control executes for producing pieces. If standard elementary entities are used, the number of entities correspond to the number of macropoints. The micropoints have to be calculated by the DSP-PCI board according to the interpolation algorithms. In order to be more precise, many users of numerical controllers prefer (for managing complex curvilinear profiles) to produce ISO program directly specifying the micropoints -- for example, when there is the needs of adopting Nurbs of Splines for the interpolations. In these cases, the numerical control has to be capable of processing a higher number of entities.

At the first glance, the ideal number of entities passed via PCI bus could be calculated by considering the bandwidth of the PCI. This evaluation is too trivial since the entities are in the order of 50 bytes and a specific protocol based on interrupt has been defined in order to synchronize the DSP-PCI board and the SIPC.

The effective measure is quite complex since it depends on: the ISO program instruction type; (ii) the maximum velocity set for moving motors; (iii) the interpolation algorithm chosen; (iv) the communication performance and mechanisms; (v) protocol for communicating between the SIPC and DSP-PCI boards, etc. All these factors cannot be measured together, thus, a distinct evaluation of the single phases is needed.

In order to perform the measure of NEPS processed by the system (and the following measures) a specific testing architecture has been defined and set: a MUPAAC Supervisor connected via Ethernet to a SIPC controlling two DSP-PCI boards. Each DSP-PCI board has been devoted to control a machine for producing passpartouts (three axes: two interpolated axes and a liner axis for the tool orientation). Even if this configuration is simple, it is sufficient to estimate all detailed execution costs.

To make the measures, a test ISO program based on 500 G1 (linear interpolation) and 500 G2 (circular interpolation) ISO instructions was realized, and sent via SIPC to both the DSP-PCI boards at the same time via the PCI bus. The same ISO program was put on execution of both machines at the same time simulating a production phase (with a velocity of 40 meters per minute). This process has been repeated 1000 times, estimating the elapsed time between the start and the completion of the program in order to have a mean estimation of the execution time. In this way, the estimation of NEPS has been possible with the precision needed. The result obtained has been of NEPS = 2556. This was a first result since the planned value was to reach at least NEPS = 1200. On the other hand, in order to evaluate the actual performance of the CNC part managed inside the DSP-PCI board several detailed issues have to be considered.

The entities are passed from the SIPC to the DSP-PCI board via PCI bus. Then the DSP-PCI has to interpret the instructions and to control the axes based on their set points. Therefore, the detailed execution times of the following phases has to be considered in order to evaluate in detail the general execution costs (see Fig.5):

(i) the communication between the SIPC and DSP-PCI boards,
(ii) the interpretation of the instructions,
(iii) the already discussed execution time, RefAx.

By using the results of a detailed performance evaluation a more precise prediction of the actual performance in other cases can be performed.

4.2.1. SIPC-DSP Communication Costs

The communication between SIPC and DSP-PCI board is mainly due to the passing of entities. These are written on the dual port memory via PCI with a double interrupt solution according to the sequence of numbers in Fig. 6.

The execution times are reported in the following table.

DSP-PCI ↔ SIPC board Communication task	worst Time (μs)	Best Time (μs)
SIPC write on DP memory (1)	17	17
DSP Interrupt latency (2)	1	1
DSP ACK write on DP memory (3)	6	6
SIPC (WIN-CE) Interrupt latency (4)	135	40
I/O cycle for R/W on SIPC (5)	120	0

In the worst case, the time for writing on the dual port by the SIPC is limited also by the other task running on the SIPC, that is devoted to the management of I/O ports via CANBUS, reported as (5) in Fig.6. From DSP the ACK is sent instantaneously at the arrival of the interrupt (2). The time needed to perform an action was measured by using *testpoints.* A digital output port has been set at the beginning of the action and reset at the end. The time needed to perform the action was externally measured with a minimum overhead. Data was collected through many executions (about 1000) and minimum and maximum values were found.

The communication between the SIPC and each PCI-DSP board takes about 279μs for every instruction on one DSP in the worst case. Thus, as 3584 instructions per second could be passed from the SIPC to one DSP. When more than one DSP are present, the time needed is expressed by STworst = (17+6+1+135) NDS+120 where NDS is the number of DSP per SIPC (ST is the Communication time).

NDS	Stworst	NEPSw	STbest	NEPSb
1	279	3584	184	5435
2	438	2283	248	4032
3	597	1675	312	3205
4	756	1322	376	2660

The values reported in the above table are referred to the worst (w) and best (b) cases. In effect, as demonstrated by the general measure, the communication performance (for a DSP with two axes) has a mean value of 2566 NEPS. This is due to the limitation of the DSP board as discussed in the following.

For improving the above performance it is possible to improve this result by sending more instructions at the same time.

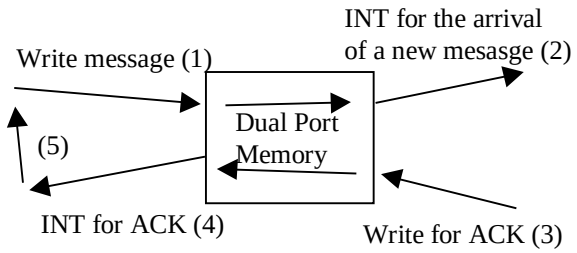

Figure 6. Detailed communication phases on PCI

4.2.2. Interpretation Costs

According to the architecture reported in Section 3.4, the interpretation includes a pre-calculus and thus depends on the interpolation used as depicted in the following table.

DSP-PCI board Interpretation task	Measured time (μs)
I/O (from CAN via PCI) and generic operations	25
Interpret. Linear Interpolation	9 + 15 per axis
Interpret. Circular Interpolation	40 per two axes

From this timing table, it can be seen that a time of 65μs is needed for interpreting an instruction for two axes in circular interpolation and 104 for 4 axes: two in linear and two in circular interpolation.

4.2.3. Discussion on NEPS.

The NEPS estimated for the whole process are due to the presence of the above costs and to the presence of queues between the components passed by the instructions from the SIPC to arrive at the control algorithm loop (see Fig.5). In particular, the above reported values for NEPS are the maximum and the minimum values that can be obtained from the SIPC part of the system, these are limited by the capability of the DSP-PCI in processing the entities.

Once defined the number of axes and considering the fixed value for the refresh cycle of motors (100μs) a small part of this is available the interpolation and communication of the instructions.

In the following table, the evolution of the execution times from some elementary configurations is reported. More complex configurations can be produced on the basis of those values. From the Table 2, if a too small number of cycles is planned (e.g., less than 7 for 4 axes) the PID is capable of processing more instructions than those that can be provided by the interpreter and the interpolator. This problems can be avoided by increasing the number of RefAx per each set point (#RefAx) and thus decreasing the NEPS managed by the DSP-PCI board.

Hence, the Remaining Time is obtained considering (100 - Actual RefAX) #RefAx.

The time needed for executing the other tasks (communication and interpretation) is fixed depending on the number of axes. It is the sum of the DSP communication time (6μs) and the interpretation time (65μs with two axes or 104μs with four axes). IDX (index) is the difference of the Remaining Time and the time for other tasks. When IDX is negative the solution is not feasible since the DSP has not time enough to do the work. In these cases, the PID is capable of processing the instructions faster than the communication and interpretation modules. Thus, it has to wait for them. This can be the cause of problems, such as the fragmentation of the motor action. This can cause large imprecision and discontinuities which may produce defects.

The NEPS(dsp) is defined as 1/(100μs #RefAx), it is the minimum rate needed by the DSP to have always instructions to execute. This rate, for each configuration, has to be provided by the SIPC. This, in turn, depends on the number of DSP boards that are present on the SIPC. Thus, NEPSw reported in the table of Section 4.2.1 has to be greater than NEPS(dsp) previously discussed. For example, the first configuration in Table 2 is not possible for two reasons, the IDX is negative and NEPS(dsp) is greater than NEPSw. Again, problems can be avoided by increasing the number of RefAx (#RefAx) and thus decreasing the NEPS managed by the DSP-PCI board.

The increment of #RefAx obviously decreases the machine velocity. On the other hand, even with values of 1 ms (10 #RefAx) the MUPAAC solution is located in the High Performance category of CNCs.

In Tab. 2, the execution CPU time is reported in micro seconds, and impossible configurations (for the first or the second reason) have been marked with * and ** in the last column.

Table 2. Configurations comparison

NA	NDS	NAD	Actual RefAx	#RefAx	Remaining Time	Other Tasks	IDX	NEPS(dsp)	Impossible
2	1	2	53	1	47	71	-24	10000	(*)/(**)
2	1	2	53	2	94	71	23	5000	(**)
2	1	2	53	3	141	71	70	3333	
2	1	2	53	4	188	71	117	2500	
2	1	2	53	5	235	71	164	2000	
4	1	4	83	4	68	110	-42	2500	(*)
4	1	4	83	5	85	110	-25	2000	(*)
4	1	4	83	6	102	110	-8	1667	(*)
4	1	4	83	7	119	110	9	1429	
4	1	4	83	8	136	110	26	1250	

(*) IDX < 0 (**) NEPS(dsp) > NEPSw (NEPSw = 3584 for NDS = 1)

4.3. Managing I/O Ports via CANBUS

The cycle of refresh of the I/O ports is the part of the MUPAAC system that, from a temporal point of view, has the lower priority. In MUPAAC, the inputs of emergency (the alarms), highly sensitive to the response time, are directly connected via the DSP-PCI board managing the axes. With this solution, for all the other I/O ports of the system it is enough to have a refresh time of 10-15 ms. This value obviously depends on the communication performance of CANBUS and, thus, a timed cycle has been set at 10 ms to perform the measures. From the measures this value has been confirmed to be large enough to perform all the I/O operations needed even in the worst case. The real cost depends on the Communication Performance via CANBUS, on the other hand, the SIPC has to use the remaining CPU time to execute the other tasks.

According to MUPAAC architecture, the SIPCs manages the control of CANBUS and the interpretation of logic equations. The estimation of the actual value for the communication performance and for the refresh time of the I/O ports have been performed by using the following typical configuration. A SIPC with (i) 64 I/O ports on CANBUS and (ii) a DSP-PCI card with 4 axes.

4.3.1. CANBUS Communication Performance

In order to stress the communication conditions (for estimating the writing time) on the CANBUS and on its peripherals: an algorithm for generating repetitive simple writing/reading on the same digital output/input was set. The communication protocol includes always a bi-directional message mechanism: writing to have an ACK back, and reading by giving an address to have the required data back. The result was 600μs for the reading/writing from/on an 8 bits digital port.

This means that a value of 13333 bps can be reached with respect to the ideal CANBUS throughput of 1 Mbps.

A similar test has been performed also for the

fragmented modality of CANBUS communication. This type of service is intended to transfer more than 6 bytes from a node to another. In this case, a time of 1200μs to transfer data on four ports (24 bits) has been obtained. In this modality a throughput of 20000 bps can be reached.

4.4. System Flexibility

The system flexibility in MUPAAC has been reached thanks to three main features:

1. CANBUS components are *plug and play*. Each I/O Module has its own identification code that can be recognized from the 8051 or SH7000 based CANBUS Peripherals. These communicate the information directly to the SIPC and from this to the MUPAAC supervisor that has in this way the exact map of physical ports available via CANBUS on the whole system, on the whole pipeline. These are mapped on logical names of variables.
2. DSP-PCI boards are *plug and play*. Thus, their presence is detected by the SIPC that in turn informs the MUPAAC Supervisor.
3. DSP-PCI boards can mange from 1 to 4 axes. According to the values reported in the above presented tables, it is possible to evaluate suitable bounds for each specific configuration of the system. In the same plant, different solutions can live together. DSP-PCI boards controlling very fast axes with other for slower axes.

5. Discussion and Conclusions

The MUPAAC architecture has been studied for satisfying the needs of production pipelines builders. Numerical controls for such a system have to be flexible and strongly expansible and reusable since the pipelines are frequently reconfigured for topology and performance requirements. MUPAAC architecture is based on a set of Industrial Computers, SIPC, connected via local area network. Each SIPC can control one or more automatic machines by means of a set of DSP-based boards. The input/outputs ports are manage via CANBUS.

The flexibility is reached by allowing the construction of a distributed control by using a variety of configurations. From the single controller to a complex pipeline with 4000 axes. During the final validation on actual pipelines for producing passpartout a relevant decrement of reconfiguration costs has been detected by VALIANI. The measures reported in this paper allows the identification of the most suitable configuration for maximizing the production velocity, or maximizing precision, or minimizing the costs. Flexible and custom configurations are also possible with some parts exploiting the maximum performance of the CNC and other with presenting low performance.

The experience reported could be of a great value for other control machine builders since describe both some interesting metrics and the process by which we have obtained the measures. The results of this assessment have been used by SED for increasing the performance of the architecture by increasing the number of entities passed from the SIPC to the DSP-board. The performance reached with the final version of MUPAAC CNC have been improved of about the 30% in certain configurations, by using 5 entities. A higher number of entities impedes the execution of the other tasks on the SIPC.

6. Bibliography

[1] F. Butera, P. Nesi. M. Perfetti, "Reengineering a Computerized Numerical Control Towards Object-Oriented", *2nd Euromicro Conf. on Software Maintenance and Reeng.*, March, 1998.

[2] C. Bruni, P. Nesi, P. Tortoli, P. Bellini and F. Guidi, "Analysis of CANBUS and PCI-based Subsystems", MUPAAC ESPRIT IV HPCN, Deliv 3.1, Jan. 1998.

[3] P. Nesi, P. Tortoli and F. Guidi, "General software with measures and examples", MUPAAC ESPRIT IV HPCN, Deliverable 4.2, August 1998.

[4] G.Bucci, P.Nesi, ``Overhead Estimation and Comparison for Multitasking Operating Systems for Personal Computers", *2nd Intern. Conf. on Parallel and Distributed Computing and Network*, PDCN'98, Australia, 1998.

[5] J.B.Chen, Y.Endo, K.Chan, D.Mazieres, A.Dias, M.Seltzer, and M.D.Smith, ``The Measured Performances of Personal Computer Operating Systems", *ACM Trans. on Comp. Sys.*, Vol14, Feb. 1996.

[6] L.McVoy, ``lmbench: Portable Tools for Performance Analysis", in *Proc. of 1994 USENIX Tech. Conf.*, USA, Jan. 1994.

[7] G.Bucci, M.Campanai, and P.Nesi, ``Tools for Specifying Real-Time Systems", *Journal of Real-Time Systems*, Vol.8, pp.117-172, March 1995.

[8] G.Bucci, M.Campanai, P.Nesi, and M.Traversi, ``An Object-Oriented Dual Language for Specifying Reactive Systems", in *Proc. of IEEE International Conference on Requirements Engineering*, ICRE'94, USA, 18-22 April 1994.

[9] F.Fioravanti, P.Nesi, S.Perlini, "A Tool for Process and Product Assessment of C++ Applications", *2nd Euromicro Conference on Software Maintenance and Reengineering*, Firenze, 8-11 Marzo, 1998

[10] D.I.Katcher, H.Arakawa, and J.K.Strosnider, ``Engineering and Analysis of Fixed Priority Schedulers", *IEEE Trans. on Soft. Eng.,* Vol.19, Sept. 1993.

[11] A.Burns, K.Tindell, and A.Wellings, ``Effective Analysis for Engineering Real Time Fixed Priority Schedulers", *IEEE Trans. on Soft. Eng.,* Vol.21, May 1995.