AIJSTPME (2011) 4(2): 1-8

A CAM controlled machine, one step to make machining as easy as printing

Didonato A.
Grenoble Institute of Technology, Grenoble, France

Reader L., Marin J., Naksangchun N., Nguyen D. N.
Responsible Design Project student, Grenoble Institute of Technology, Grenoble, France

Vignat F.
G-SCOP Laboratory, Grenoble Institute of Technology, Grenoble, France

Abstract
Manufacturing parts using a CNC machine needs highly skilled people. The development of new, user friendly human machine interfaces should make machining a designed part to be as simple as printing a document. A step towards this goal is to integrate CAM Software and CNC Machine controllers. The project aims to develop a new type of milling machine to simplify the stages between the design stage in CAD software and the manufacture of the designed part using the CNC machine. The solution chosen to achieve this is to integrate the CAM software, for this project the software is ESPRIT, with the CNC controller of the CNC machine. This project is a student project launched within the framework of an international semester project (Responsible Design) by a team of 4 students coming from all over the world.

Keywords*: CAM, Machining, Control*

1 Introduction

Manufacturing parts using a CNC machine needs highly skilled people. The development of new, user friendly human machine interfaces should make machining a designed part as simple as printing a document and reduce the human interaction required between design and manufacturing. A step towards this goal is to integrate CAM Software and CNC Machine controllers. The project aims to develop a new type of milling machine to simplify the stages between the design stage in CAD software and the manufacture of the designed part using the CNC machine. The solution chosen to achieve this is to integrate the CAM software, for this project the software is ESPRIT, with the CNC controller of the CNC machine. This project is a student project launched within the framework of an international semester project (Responsible Design) by a team of 4 students coming from all over the world. The project has been carried out in conjunction with DP technology, a leading developer and supplier of computer-aided manufacturing software for a full range of machine tool applications. ESPRIT is DP technology's flagship product and is a programming system used for milling, turning, wire EDM and multi tasking machine tools. It is the CAM software that is integrated into the system designed within this project.

2 Previous work on the subject

Some works have already been conducted to override the use of ISO language to program CNC machines. Many of them are based on STEP NC standard.

[1],[2] proposes STEP-NC Platform for Advanced and Intelligent Programming (SPAIM). This platform controls current industrial machine tools directly from STEP-NC files, which benefits from this new data model. In fact, the STEP-NC file contains all the information for manufacture, through the description of machining entities, working steps, work plane, tools, machining strategies, etc.

[3-6] proposes the use of STEP-NC to support distributed interoperable intelligent manufacturing through global networking with autonomous manufacturing workstations with STEP compliant data interpretation, intelligent part program

generation, diagnostics and maintenance, monitoring and job production scheduling.

[7] proposes an innovative language, the 'Base Numerical Control Language (BNCL),' which is based on a low-level simple instruction set-like approach. The architecture is designed around two concepts: the BNCL virtual machine, which acts as a virtual microprocessor, and the BNCL virtual hardware, which is an abstraction of the machine tool. The areas of research described above all aim to replace G-Code with a more evolved language able to address more information. However, all of them rely on the use of an intermediate language between the CAM software and the machine itself. Our proposal is to contribute to a new method with the aim of removing this intermediate file and its manipulation.

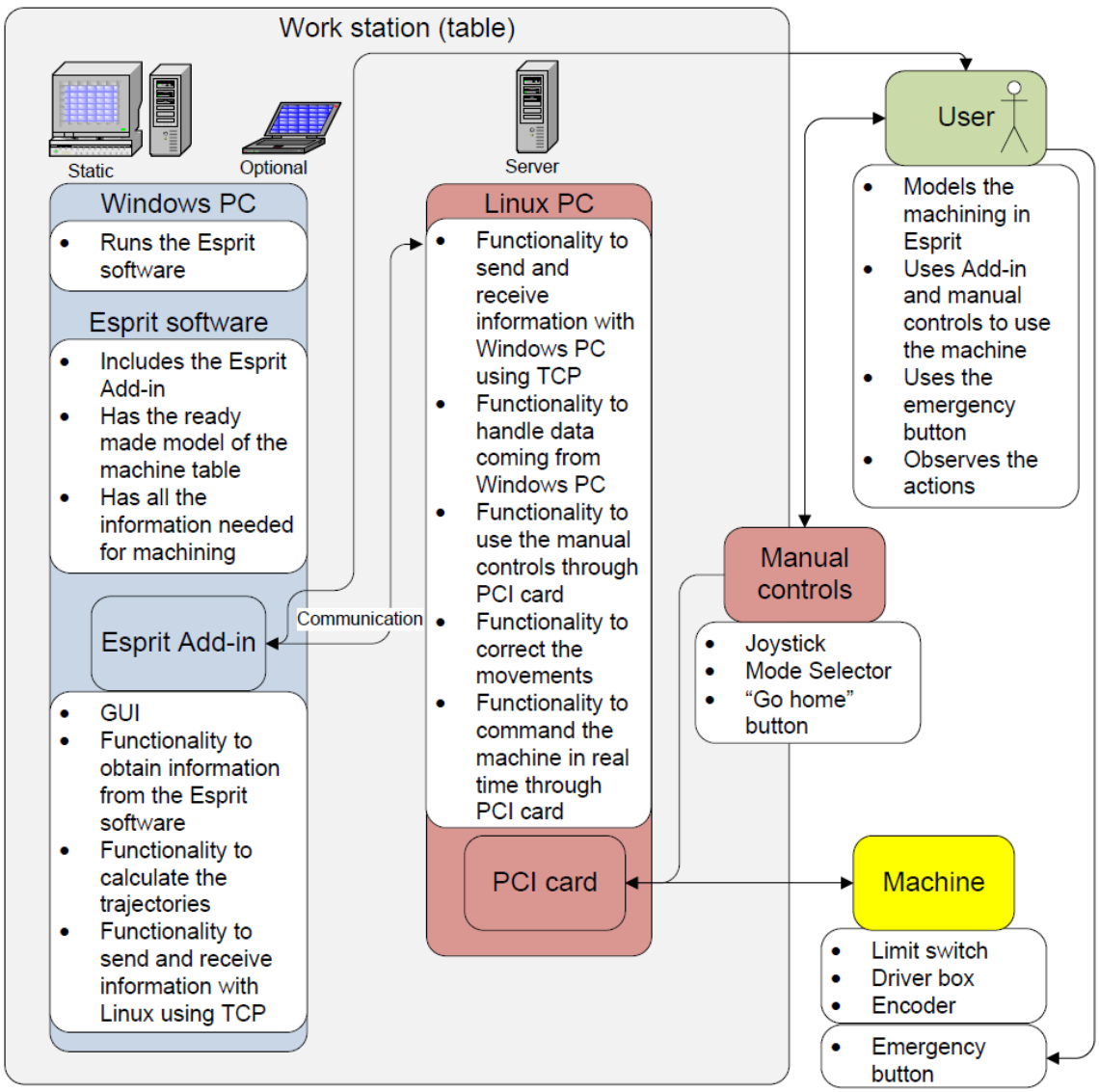

Figure 1 : Machine architecture

3 Machine architecture

The architecture of the machine is not original from an operative point of view. The mechanical part of the machine comes from a classical 3 axis milling machine. From the command point of view, the system is composed of two PC's (see figure 1). The machine is controlled by a real time Linux PC let's call it Linux PC. This PC receives instructions from ESRPIT installed on another PC trough a TCP/IP socket. This ESPRIT PC acts as a human machine interface whilst the real time PC performs without direct control. The user controls the machine via a manual control pad or via ESPRIT. In ESPRIT, the user can prepare the machining operation as usual. To start the machining operation they do not need to create an ISO file and transfer it to the CNC. They simply connect the socket to the Linux PC and click the start machining button as shown figure 2. Of course the user will have prepared the machine as usual placing the part and the tools in the right position.

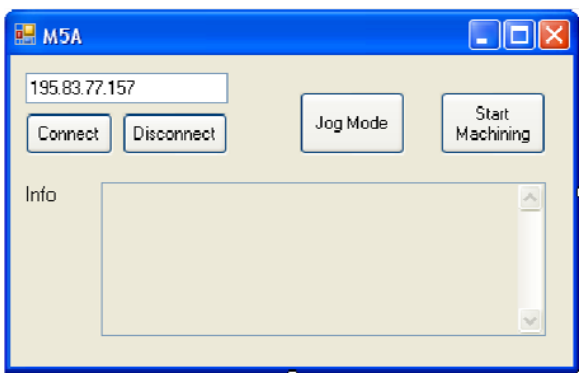

Figure 2 : Start machining window

4 What ESPRIT Does

From the ESPRIT PC side, the Esprit add-in has a major role in the system. It is able to access the Esprit document using the Esprit API. The add-in is programmed with VB.Net, it calculates tool path trajectories using this retrieved information, sends these values to Linux for operating the machine and serves as a GUI to user. The Esprit add in developed for the project consists of eight classes that are Main.vb, Ui.vb, Jog.vb, Machining.vb, Calc.vb, Waiting.vb, SendClient.vb and ReceiveClient.vb. Main class is the class that has links to ESPRIT and serves as the class that the execution starts when add in is launched. UI, Jog, Machining and Waiting serve as GUI classes. UI also has central position in the program as it handles the majority of the functioning of the program and has several ties to the other classes. SendClient and ReceiveClient are tied to the UI class and are used to communicate with Linux. Calc is also tied to UI and it's used to handle most of the actual trajectory calculations and to find the specific instance for generic technology and tool objects as well as obtain the relevant information from these. See Figure 3 for a diagrammatical representation of these classes.

4.1 *Trajectory calculation*

A major aspect of the project was the trajectory planning. Trajectory and path planning is required in order to control the machine and follow the correct paths to create the designed shape. The paths could be taken straight from the ESPRIT document, but because there is the need to include acceleration and deceleration in the tool paths and to give enough information to the Linux PC to control the machine at each time loop, the trajectory planning was required. The calculation is not presented here as far as it is similar to the one used in classical milling machines. The result of the calculation is a table describing the required speed and position required for each axis at each time loop. This is what we can call low level orders that will be sent from the ESPRIT PC to the Linux PC for machine control. Thus all the calculations needed for trajectory planning are performed in the ESPRIT PC and the Linux PC will only have to manage the machine control.

4.2 *Data communication*

For the communication between the two PCs a TCP connection with sockets is used. The TCP was selected to be used for the communication because it delivers all the messages sent trustfully (unlike for example UDP). Since communication both ways is necessary (from Windows to Linux and vice versa), both computers have server and client. Each server will create a socket, assign it a name (provide an IP address and a port to communicate), and wait for client to connect to socket. Client also creates a socket and connects to the name of socket on the server. When the server detects a connection request from a client, it will create a new socket and use it to communicate with the client. The old socket continues to wait for other connections from other clients.

In our solution there is the need for communication between the Windows PC (ESPRIT addin) and the Linux PC. In order compute the messages sent between these two PCs, a specialised syntax had to be developed for this communication. Both systems will send and receive information so the syntax has to cover the messaging in both ways. In Windows it is possible to receive complete lines so the syntax for the communication in this way is relatively simple. In Linux however, only an array of bytes is received and the messages must be constructed again. This has strong effect on the syntax, for example at the end of each line a specific mark (@) must be sent. The syntax of information sent to the Linux PC from Windows is covered here:

- To inform about the change to the auto mode "A@" is sent.
- To inform about the change to jog mode, letter "J", letter of axis ("X|Y|Z") and "@" is sent. For example "JX@" for change to jog mode in X axis.

To inform Linux about the start of whole machining "M1@" is sent to the Linux. "M0@" is sent for the end of the machining.

- Before the start of each operation (or set of toolpaths) we send letter "P", number "1" if coolant is to be used during this operation or "0" is not and "@" to mark that line end.
- For spindle speed change both in auto mode and jog mode we send letter "S", new value of wanted speed as integer between 0 and 255, and "@" to mark the line end. For example "S150@". Sent values will turn the spindle only clockwise.
- For the movement information we will simply send "FeedrateX PosX FeedrateY PosY FeedrateZ PosZ@", where FeedrateX is the feedrate of X axis and PosX the position of X axis and so on. Values of feedrate are integer values between -255 and 255. Positive value moves the machine to the direction of positive X axis. The position value is also given as integer as pulses (there are 1000 pulses in 1 millimetre). The different values are separated with whitespace.
- In the jog mode for feedrate change "F", new value of wanted feedrate as integer between 0 and 255, and "@" are sent. For example "F150@" is sent for changing the feedrate to 150. This feedrate is used with all of the axis in jog mode.
- For tool change, letter "T", the number of used tool, whitespace, length of tool and "@" are sent. For example "T1 34@". The number of tool is the same that in the Esprit, basically integer value. The tool length is an integer value in millimetres.
- For the request of machine to go home position, "H@" is sent

The information sent from Linux to Windows will also be coded using different letters for different commands (or requests). The used syntax is:

- "E" for emergency stop and "1" or "0" for the start or stop of the emergency stop. (i.e. "E1" for the start of the emergency stop)
- "A" is used for informing that the mode currently used is "Auto mode"
- "J" and "X/Y/Z" for jog mode in each axis. For example "JY" for changing to the jog mode in Y axis.

For the information about the position in jog mode or during machining Linux sends information of all axis every time the joystick stops (only once). This information is sent in the form "PosX PosY PosZ". For example "45334 34623 54934". The sent values are in pulses (there are 1000 pulses in 1 millimetre) and separated by whitespace.

- "H" is sent for telling that the machine is in the home position.

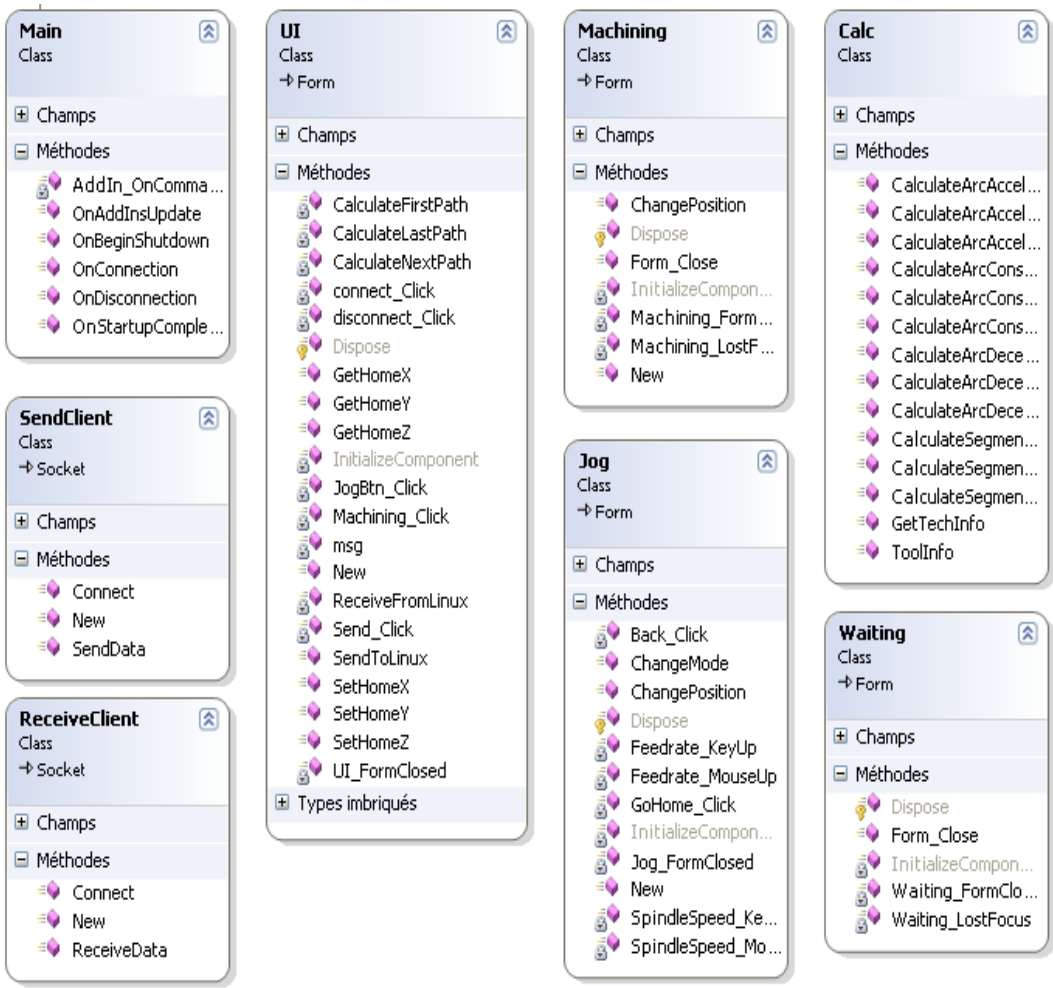

Figure 3 : AddIn composition

5 What the Linux PC does

The Linux PC runs under Xenomai [8]. Xenomai is a real-time development framework co-operating with the Linux kernel, in order to provide a pervasive, interface-agnostic, hard real-time support to user-space applications, seamlessly integrated into the GNU/Linux environment. This PC controls three programs that in three separate threads and communicate through real time pipes. Within these three threads, one is real time while two are non real time (see figure 4). The non real time program 'server_recv' receives data from Windows. This data is sent by the Windows PC using the syntax defined in section 4.2. The program then sends the data to the real time program using a real time pipe. The second non real time program 'server_send' receives data from the real time program through a real time pipe and sends it to the Windows PC. These two programs are used to transform non real time communication task between the esprit PC and the linux PC into real time communication tasks. The real time program (titled 'main task') is the main program to control the machine. At each time loop it receives the speed and position setpoint from the server_receive program. It then reads the current position and adjusts the speed setpoint function of the difference between the position setpoint and the current position. It then writes on the I/O register of the P4CNC card the adjusted speed setpoint. This program also manages the spindle speed, the tool change in automatic mode and any operations that occur in manual mode.

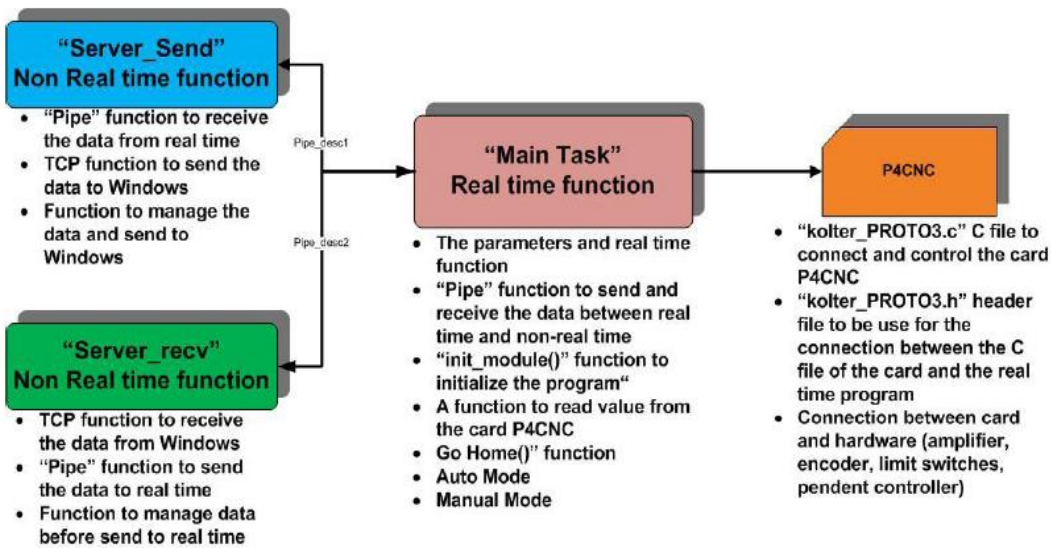

Figure 4 : Linux PC main tasks

6 Ongoing machining operation

In the ongoing machining phase, the operation runs as follows:

(1) First the Esprit add-in sends Linux "M1@" for informing that machining is started.

(2) Linux then blocks the use of jog mode. This means even the jog mode is turned on from the mode selector, Linux will not go to jog mode, it ignores the change at this point. After this, Linux waits for more information coming from Windows.

(3) During the ongoing machining the add-in will all the time get information from Esprit one operation at a time. It will calculate the trajectories for movement (basically speeds and position) and

(4) show progression of the whole machining process.

(5) The add-in will send letter "P" and number 0 or 1 and "@" to Linux in order to inform about the starting of the machining. The number 0 is send for not using coolant during this operation and 1 is send for using. "@" is just used to indicate the line end.

(6) If Linux receives "P1" it turns the coolant on. Also the Linux will interpret the upcoming lines as information for movements.

(7) Before starting operation, add-in checks the technical information of the operation. If tool change is needed it's also done at this point. In the tool change the tool length is also compensated to the home positions. This procedure is described in the "Tool change" part

(8) After possible tool change spindle speed for operation is send to Linux. For this "S" and spindle speed as integer from 0 to 255 and "@" is send (for example speed 150 "S150@" is send). Originally speeds were designed to go from -255 to 255, negative meaning counter clockwise movement, but in this solution we only have clockwise movement.

(9) The spindle speed is changed in Linux.

(10) After this Esprit add-in sends the movements line by line. Used form is "feedrateX positionX feedrateY positionY feedrateZ positionZ@". The values for tell the current feedrate and position for each axis in this point of machining. For the feedrate of each axis integer value from -255 to 255 is used. Position can be integer from the whole range, but in reality the values are limited to the positions that the machine can move to.

(11) In each loop the Linux also reads the position of the machine from the encoder.

(12) It then uses this value as a reference to a value obtained from the add-in and makes corrections to the feedrates if the positions do not match.

(13) Then the Linux commands the machine using the corrected feedrates.

(14) This moves machine, or actually changes the feedrates of each axis. The commanding of the

machine is done in real time using the Xenomai. The information between real time and non-real time is transferred using "pipe" functions of Xenomai. This function is kind of a buffer.

(15) After the last line describing the movement in operation, "#@" is sent to Linux to inform that it is the end of that operation.

(16) When Linux receives this mark, it will turn the coolant off and start to interpret the upcoming lines as different kinds of commands.

(17) After finishing the whole machining process Esprit add-in still sends "M0@" to Linux to describe the end of machining operation.

(18) When Linux receives this it will stop the spindle by putting the speed value to 0. NOTE! During the whole machining tasks 3-16 are repeated for each operation. Inside one operation tasks 3, 4 & 10-14 are repeated for continuous movement.

7.Conclusions

This paper presents a new type of CNC system that aims to simplify the stages between the design stage in CAD software and the manufacture of the designed part using the CNC machine. The solution chosen to achieve this is to integrate the CAM software, for this project the software is ESPRIT, with the CNC controller of the CNC machine. The architecture of the CNC is based on the use of two PC's. The first PC runs ESPRIT under a Windows operating system. It manages the human machine interface and the tool path calculations. It then sends, through a TCP socket, the information concerning the tool path incremented with a time step of 1 ms. The second PC runs Xenomai real time framework under a Linux operating system. It manages the machine control. This new type of CNC simplifies the use of the machine allows for direct control of machining through the ESPRIT software. The user still requires the knowledge to prepare the machine properly; putting part and tools in the right position. Some works using vision and recognition algorithms [9] should allow going further in this direction.

Acknowledgments

The authors would like to thank DP Technology for their support. They would also like to thank all the associates of the ginova platform and especially Jean-François genestter and Thierry hennocque for their kind and strong support.

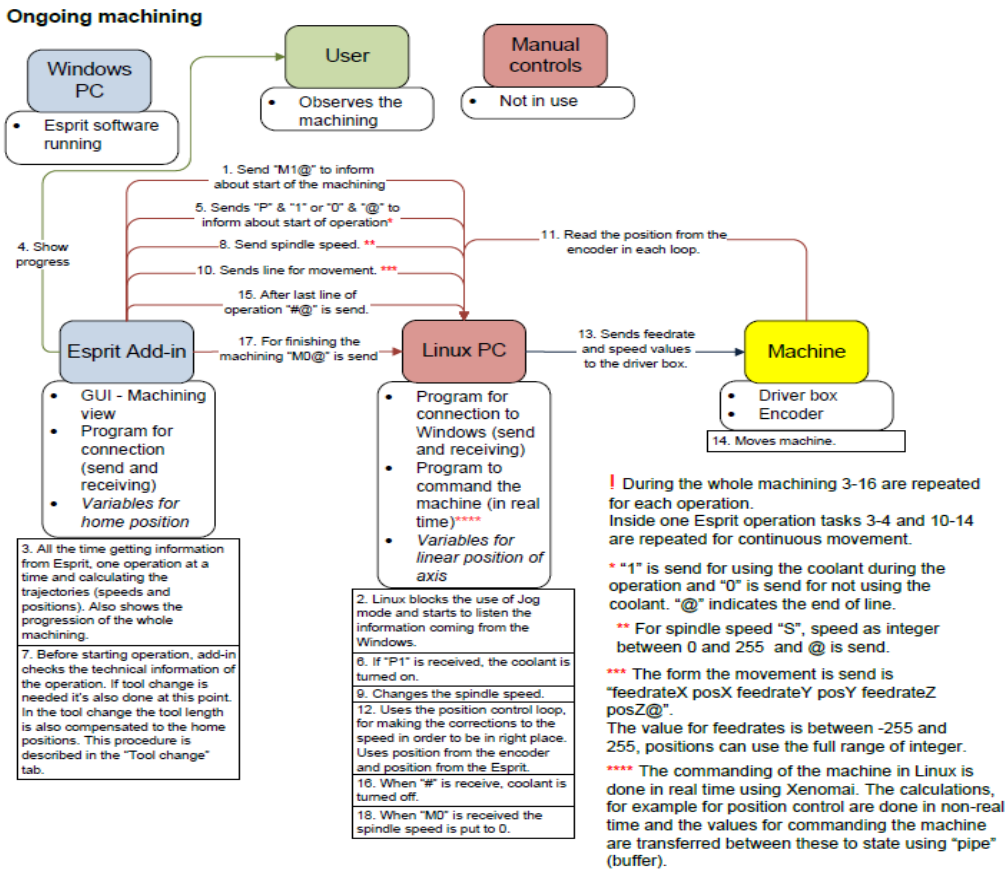

Figure 5 : Ongoing machining operations

References

[1] Laguionie R., Rauch M. et Hascoët J.Y., 2009. Toolpaths Programming in an Intelligent Step-NC Manufacturing Context, Arxiv preprint arXiv:0905.3079,

[2] Rauch M., Laguionie R. et Hascoët J.Y., 2009. Achieving a STEP-NC Enabled Advanced NC Programming Environment, Advanced Design and Manufacturing Based on STEP, 197–214.

[3] Newman S.T., Allen R.D. et Rosso R.S.U., 2003. CAD/CAM solutions for STEP-compliant CNC manufacture, International Journal of Computer Integrated Manufacturing, 16(7): 590–597.

[4] Xu X.W. et Newman S.T., 2006. Making CNC machine tools more open, interoperable and intelligent—a review of the technologies, Computers in Industry, 57141–152.

[5] Xu X.W., Wang H., Mao J., Newman S.T., Kramer T.R., Proctor F.M. et Michaloski J.L., 2005. STEP-compliant NC research: the search for intelligent CAD/CAPP/CAM/CNC integration, International Journal of Production Research, 43(17): 3703–3743.

[6] Xu X.W. et He Q., 2004. Striving for a total integration of CAD, CAPP, CAM and CNC, Robotics and Computer-Integrated Manufacturing, 20(2): 101–109.

[7] Fortin E., Chatelain J.F. et Rivest L., 2004. An innovative software architecture to improve information flow from CAM to CNC, Computers & Industrial Engineering, 46(4): 655–667.

[8] http://www.xenomai.org,

[9] Zhang X., Tian X. et Yamazaki K., 2010. On-machine 3D vision system for machining setup modeling, The International Journal of Advanced Manufacturing Technology, 48(1): 251-265.