A PLC-based Implementing Framework for Order Holon Execution in HMES

Theodor Borangiu, Nick Andrei Ivănescu, Andrei Roşu, and Mihai Pârlea.

University Politehnica of Bucharest, spl. Independentei 313, Romania
Tel: +40 214029314; E-mail: borangiu@cimr.pub.ro, nik@cimr.pub.ro, andrei@cimr.pub.ro, mihai.parlea@cimr.pub.ro
URL: http://www.cimr.pub.ro

Abstract. The paper describes a modern solution for controlling a complex manufacturing cell, consisting of conveyors, robots, stoppers and other devices, using a single programmable logical controller (PLC). A special array of data structures is used to easily access and manage the pallets processed at the four robotized stations of the cell. Also, high performance communication protocols are implemented in order to allow PLC – Robot interaction, protocols that use both I/O lines and TCP/IP.

Keywords. Intelligent manufacturing system, holonic manufacturing system, flexible and reconfigurable manufacturing system, production activity control, supervisory control, PLC.

1. Introduction

Traditional networked assembly structures have either a hybrid or heterarchical architecture. The first type allows data exchange and co-operation between lower-level (robot) controllers. In this architecture, a master initiates all the activities and then the subordinates cooperate to perform them.

The second type of architecture is formed by a group of independent entities, usually called agents, that bid for orders based on their status and future workload (Morel et al., 2003). The master-slave relationship is dismissed and due to this decentralized control architecture, the agents have a complete local autonomy and the system is able to react promptly to any event such as: resource failure, new customer order etc (Rahimifard, 2004). Global batch optimization it is however improbable, because the execution of one order depends on the features of other orders, and also the system's performance is unpredictable.

In order to face resource break-downs, job shop assembly structures use networked robot controllers with multiple-LAN communication facilities allowing for production data saving and automatic re planning of batch production by help of failure and recovery management (Babiceanu et al., 2007).

The manufacturing cell in which this project was implemented consists of six robots and a conveyor system. The robots are able to communicate with the PLC by the means of digital Input/Output lines and also by the means of Ethernet TCP/IP, being able to exchange information about their status and information regarding the manufacturing process of the pallet that is in the post. One Cartesian robot is used for inserting pallets in the system. The stoppers existent in the system can block or not the movement of the pallets on the conveyors, four of them being related to code-read sensor and other two to a read/write code sensor heads. Lifts have the task of transferring the pallets from one main conveyor belt to another or from the main conveyor loop to the robot conveyor belts that take the pallet to the working stations. The conveyor system is controlled by a PLC that enables all mechanical actions to be taken in the transport system (Lastra and Delamerm, 2006).

The PLC, as seen in Fig 1, is the core component of the entire cell. It has to take the data necessary for production from a planner. Then needs to control al the mechanical elements of the transport system, has to monitor and track pallets throughout the system by the means of an identification system and finally to communicate and give commands to the executing robots.

Pallets enter the system having the possibility of being written with a unique magnetic-code for identification during the process. Each pallet, according to a previous planned production, stops at one or more working stations in order to have the operations done. Once all operations are complete, the pallet exits the system. The maximum number of

pallets that can enter the system, for a batch, is 256, limited by the eight bits available for the magnetic code. For practical reasons, each pallet can support a total of 16 operations. An offline planner has to create an optimal schedule by maximizing the load of all available machines.

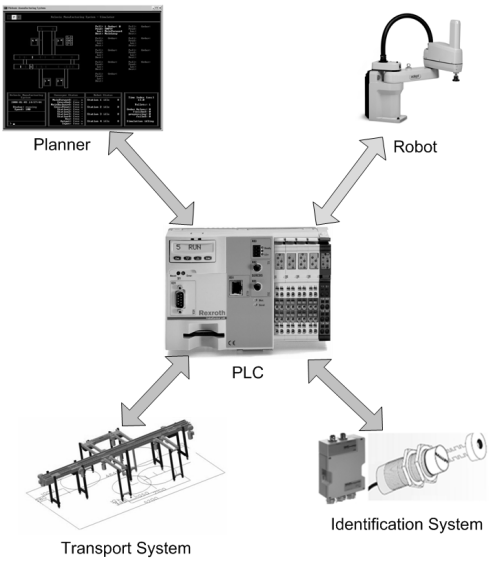

Figure 1 – The main configuration

When the products enter the system their memory capsule is written with their unique code for identification throughout the production of the batch. The products are then transported to the next working station by advancing on the main conveyor loop. At the five locations where they can be deviated there are sensors that identify the unique code, and according to the data received from the planner will take the actions for the deviation if necessary.

The robots present in the manufacturing system should have a minimum of idle time, although in some particular cases some of the robots may not be involved. For achieving a maximum load, the conveyor system should never be blocked by any product that is waiting to be processed by a robot. If the transport system is not overloaded, the robot station can always be reached without waiting times and can always carry out a task.

Some constraints are applied to the system:

- at any time in the system there can be no more then 4 products.
- the products coming out of a post have priority over the ones moving on the main conveyor.

2. Solution design and PLC-based implementing of order holon

2.1. Theoretical backgrounds

When dealing with complex systems having a big number of input and output signals having to be controlled by a single PLC, a modern solution for software design must take into consideration the following issues:

PLC must be chosen such as to support advanced software development kit, containing most of the standard programming languages for PLC's, like Sequential Function Chart (SFC), Ladder Diagram (LD) or Structure Text (ST)

The project should contain both graphical sequential programs and cyclic programs (for actions needed to be performed at every PLC cycle)

In most of the cases the project should contain several programs that will run simultaneously, every program managing usually one action device (like relays, actuators and so on) or managing other activities like communication or timing management.

The idea is not to use a centralized type of control, with one big "parent" program trying to implement all the actions, but to use several smaller programs, communicating between each other by global variables, each one taking care of a small part of the process.

2.2. Project structure

Following the guidelines previously described, a Bosch PLC together with Indralogic development software was used to accomplish the task. The project's structure is shown in fig. 2. The project consists of 34 programs and 3 functions. All these can be divided into 9 categories:

- Stopper
- Lift
- Transfer
- Communication with Robot (I/O)
- TCP data send to Robot
- Write Code on Product
- Time Keeper
- User Interface
- OPC Communication

The "Stopper" type program has as objective the control of the mechanical device with the same name. Its function is to stop verify that if the next segment where the product needs to go is available or not, and activate or not the stopper. Also the program has to deviate the product towards the working station of a robot if necessary.

The "Lift" type program has as objective the control of the mechanical device with the same name. Its function is to lift the product from the level of the main conveyor belts to the level of the lateral conveyors of each robot working post. Once the product has reached the working place the program signals the robot for the beginning of the necessary operation.

The "Transfer" type program must control a double lift and the preceding stopper, its function being to transfer the product from one main conveyor belt to the other.

The "Communication with Robot" type program implements a query protocol over the digital

input/output lines for all the resource controllers and has to decide whether the controller is working properly and to communicate the status.

The “TCP data send to Robot” type program has to send over the Ethernet a request for a job to a robot working station and must wait for the answer using the three functions for the TCP communication.

The “Write Code on Product” type program has to write the unique identification code on the product memory capsule. The code is 8 bits long and in order for it to be valid the binary one’s complement must be written at the next two memory addresses.

The “Time Keeper” type program must increment, with a step of 500ms, an integer variable that can communicate the time of production. As a second function, the program must reinitialize the system when a batch is finished.

The “User Interface” type program has to control some of the graphical user interface elements, and to show the data in the system in a easy to understand format.

The “OPC Communication” type program must control the data flow from the planner to the transport system.

One of the issues in solving the transport system control problem was the entry. Due to the fact that products enter the system at different time stamps, and other may need to be transferred from one main conveyor belt to the other, some kind of interblocking must be implemented. The first solution one might take in account is a simple TAS (Test And Set). The product that needs to enter the system tests to see if the transfer is clear, signals the other products interested that the region has been occupied by setting a variable, and takes the necessary mechanical actions to make the product enter the system and when it has finished resets the variable, clearing the transfer. If by any chance the two products make the test in the same time then they are both going to occupy the transfer and a crash is possible. The solution is to prior test the sensors that detect the products and give priority to the one that needs to be transferred, because it must reach another working post.

When one product is exiting a working post and another product that is on the main conveyor would like to cross to the next point, then a priority management must be implemented. It was decided that the exiting product has a higher priority then the products on the main conveyor loops.

Another special issue is accessing the data structure. In order to have an easier access to the data it was decided to use an array that points to the next operation that the product must bear.

The application has to convert the data coming from the planner that it is used (unlike a product driven approach (Petin et al., 2007)) into mechanical movements of its constituent elements and, the first step in solving the control problem is to choose the way in which this data is stored (Borangiu et al., 2008). Having taken into account the way a product is developed it was decided to use this structure:

```
TYPE datemasina :
STRUCT
post:BYTE; (*number of the robot working post*)
operatie:BYTE;(* a code repre-senting the operation done at this post*)
timpmin:WORD;(*the minimum amount of time necessary for completion *)
timpmax:WORD;(*maximum amount of time for completion of the operation*)
raport:BYTE;(*a small report about the result of the operation*)
END_STRUCT
END_TYPE
```

This structure is repeated for each operation that the product has to pass and these structured compose an array named “sir_palete”. It was decided that 16 structures are sufficient to completely describe a product and due to limitations of the system there can be no more then 256 products in one batch, these being the reasons why “sir_palete” was defined as an array of 256 by 16 structures of type “datemasina”. In order to have access to the data in the array one needs two indexes: the first index is the product number (0-255) and the second is the number of operation (1-16). The array named “sir_index” has a length of 256 fields each of them being the operation index reached for the corresponding product.

The data structure necessary for the information regarding the production is very large 256 (products) x 16 (operations) x 5 (elements in a structure) = 20480 items and the limit imposed by the OPC standard is 15000 (1,5 Mb). For this reason it was decided to send one product at a time over the OPC, meaning 16 structures of 5 elements. For the control of the communication another two elements were necessary one that determines the kind of action (read/write) and another for synchronizing the planner and the PLC.

```
post1: BYTE;
    operatie1: BYTE;
    timpmin1: WORD;
    timpmax1: WORD;
    raport1: BYTE;

post2: BYTE;
    operatie2: BYTE;
    timpmin3: WORD;
    timpmax3: WORD;
    raport3: BYTE;
……..
post16: BYTE;
    operatie16: BYTE;
    timpmin16: WORD;
    timpmax16: WORD;
    raport16: BYTE;
```

In order to write these in the PLC the planner must first write in the synchronization variable the code of the product for which the structures are valid

and then give the write command. After 256 cycles the whole structure is written. Accompanying this data is also a string that shows the moment of time the product should enter the system. These are kept in an array "time_insertion".

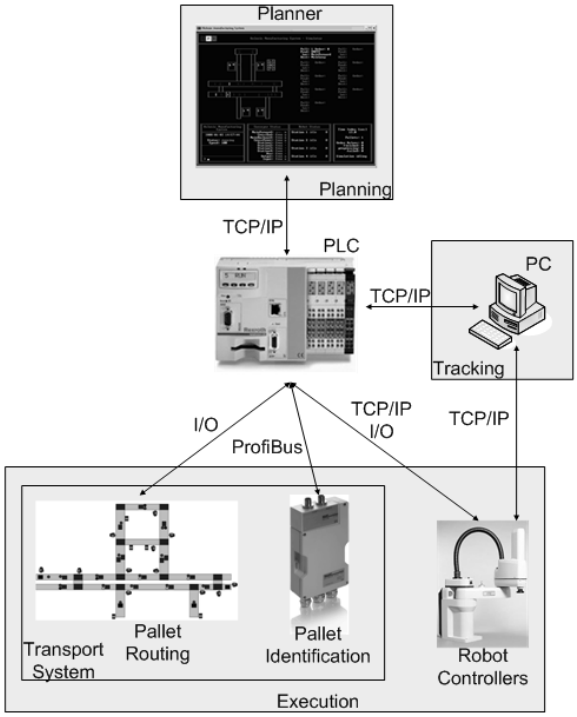

Figure 2 – PLC Communication Interfaces

3. PLC - Robot Communication

The PLC is the key element of the cell because it is the central node that facilitates the communication between all holons involved and has the task of executing the transport operations involved.

So, the communication between PLC and Robots is based on a dialog type model. According to this model, any communication protocol is initiated by the PLC, the Robot executes the orders and answers only in the mode depicted by the protocols.

Any communication protocols that will be implemented will have three main tasks:

- to monitor the robot's online / offline status
- to coordinate the robot task execution
- to transmit the codes corresponding to the requested job and complementary execution report

In order to implement such protocols, we have two communication interfaces at our disposal: I/O lines and TCP/IP (see Fig. 2).

The I/O lines are direct links between each robot and the PLC. These are binary lines of industrial 24VCC level, perfect for signaling different states and launching operations. On the other side, these lines have a high cost and are not very well suited to transmit data (through an eventual parallel protocol).

The PLC and all the Robots are connected to a Switch through the TCP/IP interface, so that data packages can be sent / received directly, the TCP/IP protocol being in charge of correct package routing. Although we are using shielded cables and the TCP/IP interfaces implemented in the PLC and Robot Controllers are of high quality, this protocol was not created to be of industrial use. So, this interface must be implemented with great care and only for data transmission.

Considering the existing requirements and the available communication interfaces, two protocols will be implemented:

- Ping – this protocol detects the online / offline status of the robots
- Synchronization – this protocol implements the robot task execution

3.1. Ping Protocol

Since it is of the upmost importance to know which robots are online and when a robot changes it's online / offline status, a protocol is implemented in order to provide this information (Barata and Camarinha, 2000). Because of the high redundancy required by this protocol, it uses only the I/O lines.

The lines used by this protocol are:

- PLC → Controller
  - Request Status
    - request the Controller to signal that he is online
- Controller → PLC
  - Acknowledge Status
    - answer from the Controller

3.2. Synchronization protocol

This protocol must implement both PLC – Robot synchronization and job / complementary report codes transmission. So, it is necessary to use both I/O lines and TCP/IP interfaces:

- PLC → Controller
  - Request_Job
    - signals the Controller that the PLC wants a job to be executed
  - Pallet_In_Position
    - signals the Controller that the PLC has brought the pallet in the working position and that job execution can be commenced
- Controller → PLC
  - Ready
    - signals the PLC that a job is in execution
  - Job_Done
    - signals the PLC that the current job has been executed
- Bidirectional (PLC ↔ Controller)
  - TCP/IP

PLC transmits the job code, the Controller transmits job acceptance report and (if the job was accepted), transmits job execution report upon job completion.

The protocol runs as follows:

- the PLC detects that Ready is 0 so it sets Request_Job to 1 and transmits the job code over TCP/IP
- the Controller reads the TCP/IP code and evaluates if it can execute the job; if he can execute the job then it will send the job acceptance code, if not it will send a job reject code; if the job is rejected then communication stops (we assume that the job is accepted)
- the PLC brings the pallet in the working position and sets Pellet_In_Position to 1
- the Controller sets Ready to 1 (the PLC sets Request_Job to 0) and begins executing the job
- upon job execution, the Controller sets Job_Done to 1, Ready to 0, and sends the job completion report over TCP/IP
- the PLC takes the pallet, sets Pellet_In_Position to 0, the Controller sets Job_Done to 0 and is ready to recommence the protocol

The protocol's evolution over time is also presented in Fig. 3.

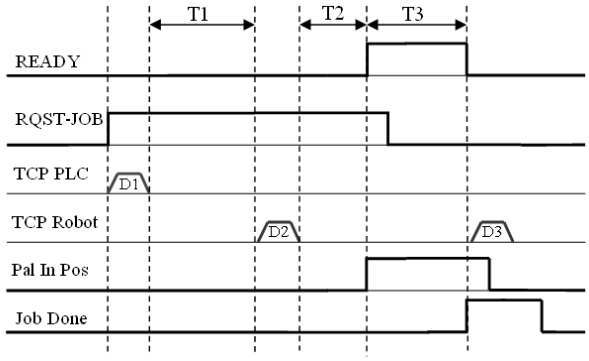

Fig. 3 – Communication signals

The three time intervals presented in Fig.3 stand for:

- T1 – interrogation time of part supply and/or workstation equipments in order to reach to a job acceptance / rejection conclusion
- T2 – pallet transportation time from the main conveyor to the workstation
- T3 – job execution time

4. Multi-tasking control of a working station

The robot controller features a multitasking industrial processor. Just like any other multitasking processor, this processor can execute a single task at any given time, but all tasks take alternatively control of the processor for very short periods of time, thus creating the impression that all tasks are running simultaneously. In order to decide which task deserves to run next on the processor, processing time is divided in a major time slices, each of these being 16ms long. Every major slice is divided in 16 equal minor slices. Each system or user task has a priority (ranging from -1 (do not run) to 63 (maximum)) assigned for each slice. At the beginning of each minor slice, the processor makes a list of ready to run tasks and assigns control over the processor to the task with the highest priority, when this task finishes running, priority is assigned to the next task and so on until all tasks run or the slice ends.

Because (without an additional license) the number of user tasks is restricted to seven, we tried to use the least number of tasks possible in order to implement the communication protocols. This way, we leave the maximum number of free tasks; these tasks will be used for future development and for controlling other workstation equipment.

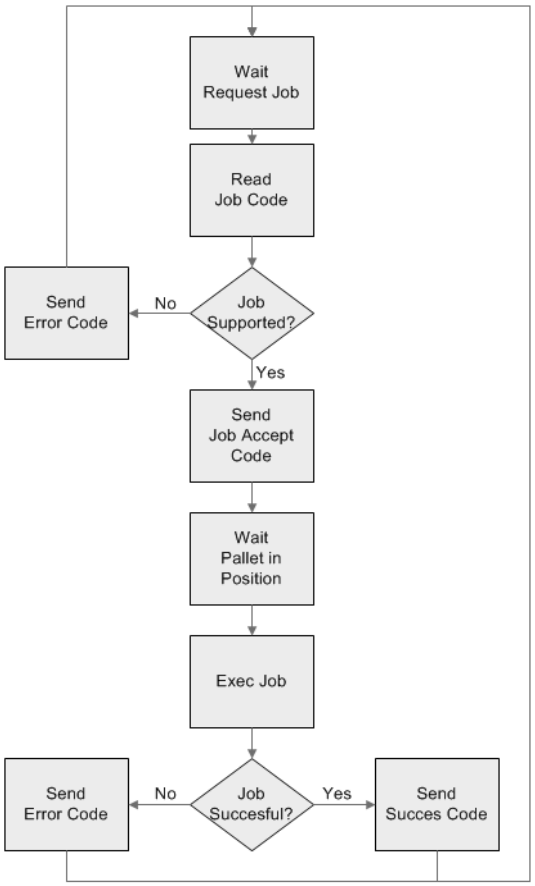

Fig. 4 – Synchronization protocol

An essential request is for the Controller is that the communication protocols are not interrupted by the robot failing various mounting operations.

It results that we need to occupy a minimum of two tasks: one task for robot operations and one task for the communication protocols.

From the controller's point of view, the Synchronization protocol implies the steps presented in Fig. 4. Also, this protocol must be executed in parallel with the Ping protocol on the same task. In order to achieve this, a WHILE loop that repeats infinitely is used, this loop executes first the Ping and then the Synchronization protocol. In order to know the current position of the Synchronization protocol for each iteration, this position is stored in a variable. At each iteration the variable is read so that the correct stage of the protocol is run, if the protocol steps to the next stage, then the variable is updated, so that the next stage is run at the next iteration.

5. Conclusions

After a theoretical evaluation of the presented concepts the conclusion was that they fulfill the requirements needed for driving the manufacturing cell.

Once implemented, it was concluded that the concepts presented successfully fulfilled the task of driving the production cell.

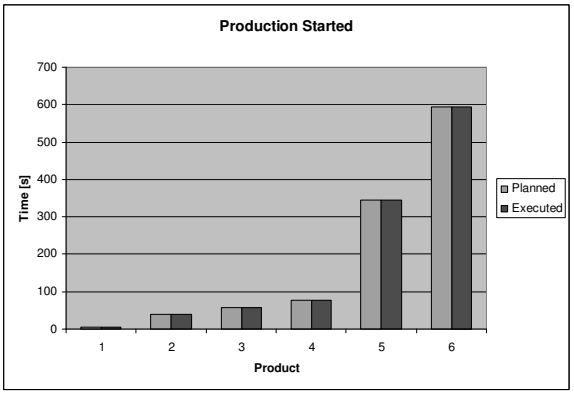

Fig. 5 – Production Started times

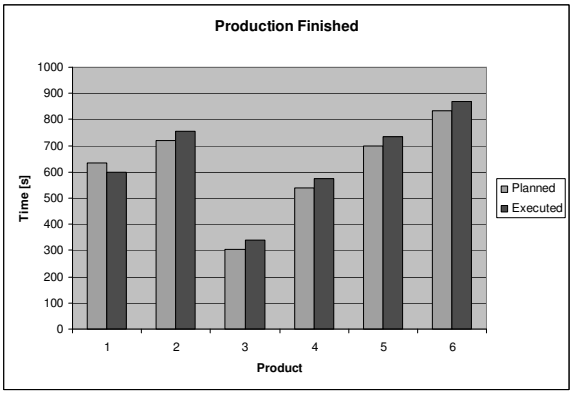

Fig. 6 – Production Finished times

On implementing control of additional equipment such as vision software, automatic feeding of bulk parts, CNC control for drilling and milling, the capacity of the multitasking structure to execute additional tasks without interrupting the PLC communication protocols was observed. Also, the performance of using only two tasks (while leaving five free tasks) was noted.

Once all the programming was done and the cell was fully functional, we programmed the PLC to report the start and finish time for each product, both the planned and the executed value. The results can be seen in Figs. 5 and 6. The conclusion is that, although there are differences for the planned and executed finish time (which will be addressed in future activities), they are small compared to the overall production time and that the system behaves as expected.

6. Acknowledgments

This work was partially supported from the scientific grant 146 / 2007 "Autonomous, intelligent robot-vision platforms for product qualifying, sorting / processing / packaging and quality inspection with **S**ervice-**O**riented, **F**eature-based **H**olon**I**c **CO**ntrol a**R**chitecture – SOFHICOR" of the National Agency of Scientific Research (ANCS).

7. References

Babiceanu, R.F. et al (2007). Framework for control of automated material-handling systems using holonic manufacturing approach, Int. J. Prod. Res., 42, 17, Taylor & Francis, 3551-3564,

Barata, J. and L.M. Camarinha-Matos (2000). Shop floor re engineering to support agility in virtual enterprise environments, in E-Business and Virtual Enterprises, Kluwer Academic Publishers, London, 287-291.

Borangiu, Th., Gilbert, G., Ivanescu, N. and A. Rosu (2008). Holonic Robot Control for Job Shop Assembly by Dynamic Simulation, *Proc. of the 16th Mediterranean Conference on Control and Automation – MED'08*, June 2008, Ajaccio.

Lastra, J. and I. Delamerm (2006). Semantic web services in factory automation: Fundamental insights and research roadmap, IEEE Trans. on Industrial Informatics, 2, 1-11.

Morel, G., Panetto, H., Zaremba, M., Mayer, F., 2003. Manufacturing enterprise control and management system engineering: Rationales and open issues, IFAC Annual Reviews in Control.

Pétin, J.-F. and G. Morel (2007). A product-driven reconfigurable control for shop floor systems, Studies in Informatics and Control, 16

Rahimifard, S., 2004. Semi-heterarchical production planning structures in the support of team-based manufacturing, International Journal of Production Research, 42, 17, September 1, 3369-3382(14), Taylor and Francis Ltd.