IPP-HURRAY! Research Group

Polytechnic Institute of Porto
School of Engineering (ISEP-IPP)

Integration of an Automatic Storage and Retrieval System (ASRS) in a Discrete-Part Automation System

Alejandro CRESPIN
Pedro CABELLO

HURRAY-TR-0421
July-2004

relatório técnico

technical report

Integration of an Automatic Storage and Retrieval System (ASRS) in a Discrete-Part Automation System

Alejandro CRESPIN, Pedro CABELLO

IPP-HURRAY! Research Group
Polytechnic Institute of Porto (ISEP-IPP)
Rua Dr. António Bernardino de Almeida, 431
4200-072 Porto
Portugal
Tel.: +351.22.8340502, Fax: +351.22.8340509
E-mail: { hastaluego44, p02mecap}@hotmail.com
http://www.hurray.isep.ipp.pt

Abstract:
This technical report describes the work carried out in a project within the ERASMUS programme. The objective of this project was the Integration of an Automatic Warehouse in a Discrete-Part Automation System. The discrete-part automation system located at the LASCRI (Critical Systems) laboratory at ISEP was extended with automatic storage and retrieval of the manufacturing parts, through the integration of an automatic warehouse and an automatic guided vehicle (AGV).

ALEJANDRO NAVARRO CRESPIN
PEDRO JOSÉ MEDINA CABELLO

Integration of an Automatic Storage and Retrieval System (ASRS) in a Discrete-Part Automation System.

UNIVERSIDAD DE CÓRDOBA
ESCUELA POLITÉCNICA SUPERIOR

E
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
DPTO. DE ENGENHARIA ELECTROTÉCNICA

JULY 2004

ALEJANDRO NAVARRO CRESPIN
PEDRO JOSÉ MEDINA CABELLO

Integration of an Automatic Storage and Retrieval System (ASRS) in a Discrete-Part Automation System.

Report of the work developed in the environment of the discipline of Project End of Studies of 3rd grade of Industrial Technical Engineering, Industrial Electronic.
Supervisor:
Mário Ferreira Alves

IPP-Hurray Research Group.
Instituto Superior de Engenharia do Porto.

July 2004

Acknowledgements.

The accomplishment of this work would not have been possible without the collaboration of many people. We would like to thank:

- The institutional support of the Polytechnic Institute of Porto and of the University of Córdoba which have negotiated the Erasmus scholarship so that we can make the project in the ISEP (Instituto Superior de Engenharia do Porto).

- We would like to thank to the department IPP-Hurray (HUgging Real-time and Reliable Architectures for computing sYstems) of the ISEP the given support, especially to our supervisor Professor Mario Ferreira Alves, Nuno Pereira, Emmanuel Lomba and Luis Miguel Marques for their dedication and help.

INDEX

1. INTRODUCTION.

1.1 Objetives.

The objective of the project is the integration of an automated warehouse AS/RS (Automatic Storage and Retrieval System) into a discrete-part automation system.This system is configured for the transport, classification and distribution of parts.

The ASRS is an automated warehouse which moves in the Cartesian Coordinated, X, Y and Z. With the help of a mechanical arm catches the pallets of the warehouse. Another part of the system is the Automatic Guided Vehicle (AGV). So, the content of the palettes of the warehouse are deposited on the AGV.

The global system includes equipment such as input/output buffers, conveyor belts, pneumatic cylinders, robot arms and automatic vehicles. The selected pieces arrive at the end of a conveyor belt, where they are deposited in the AGV, with the rest of the pieces of the warehouse.

Lastly, robot arms are used to unload the buffers from the AGVs (unload station). There are stations to control the AGV in the warehouse, under the conveyor belt and next to the unload station.

The goal of the project is to control the warehouse and the AGV through the work station (PC), establishing a communication protocol between the work station and these devices (ASRS and AGV).

1.2 Structure of the Report.

The report is structured in the next way:

At the beginning we will make a description of the manufacturing system where the warehouses are included, and we will explain the operation of the global system.

Next we will study the integration possibilities and communication of the warehouse and of the AGV with the work station and we will make a justified decision.

Later on we will deepen more over the ASRS. Hardware, software, operation. The following step will be the study and operation of the AGV.

Next we will develop an application to communicate the ASRS for the serial port of the PC and the AGV for the parallel port.

We will continue with the use of the tool WinCC of Siemens to develop an application in which the warehouse is integrated in the system. Lastly we will expose the conclusions.

2. THE MANUFACTURING SYSTEM.

2.1 Manufacturing System Operation.

The existing manufacturing system consists on a group of processing machines or workstations interconnected by an automated material handling system and operated as an integrated system under computer control. The layout of the manufacturing application is presented in the image below, in the figure 2.1.

When a new part arrives (is transported to this subsystem), it must be classified according to a certain criteria and must be distributed to storage buffers or to the next stage of the manufacturing process. This next stage could be further processing (cutting, drilling, etc.) or just transporting a storage buffer to a warehouse. Roller belts and different pneumatic equipment are used to transport and distribute parts to output buffers, according to their type. When output buffers are full, they are moved (either by an automatic vehicle, a robot arm, or an operator) to the respective unload station, in order to be emptied. Considering the classification criteria, at this stage each part is distinguished by its colour.

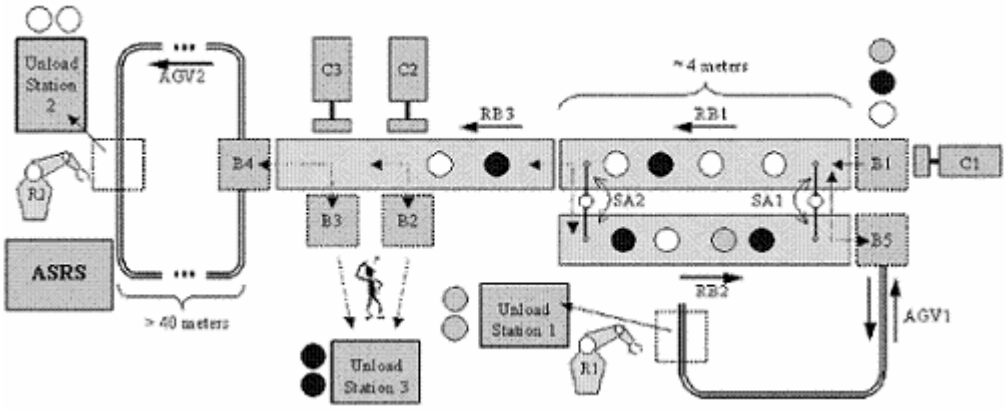


Figure 2.1 Layout of the Manufacturing System.

The input buffer (B1) stores black, white and grey (defective) parts, which are sequentially pushed into the roller belt (RB1). SA2 (a swivelling double arm with suction cups) pushes grey parts to RB2. Grey parts go into B5. If this buffer is full or in transit grey parts must circulate around RB1-RB2. When B5 is full, AGV1 moves to U1, for unload operation carried out by a robot arm (R1) and an operator, and then returns to the initial position. White and black parts go into RB3, and black parts are pushed into output buffer (B2). When B2 is full, an operator is warned, in order to unload it. Meanwhile B3 must be used to receive black parts. If both B2 and B3 are non-operational (full or in transit), black parts must circulate in RB1-RB2. White parts go into B4, until it is full or if it is in transit. When B4 is full, AGV2 moves to U2 with pieces of the warehouse and the belt, for unload operation carried out by R2. White parts must circulate around RB1-RB2, if B4 is unavailable. We must integrate in the global system the automatic warehouse ASRS and the AGV 2 [3].

2.2 Integration of ASRS.

For the integration of the warehouse we should control it by means of our work station establishing a communication protocol between both. In the same way we should communicate the work station with control station of AGV.

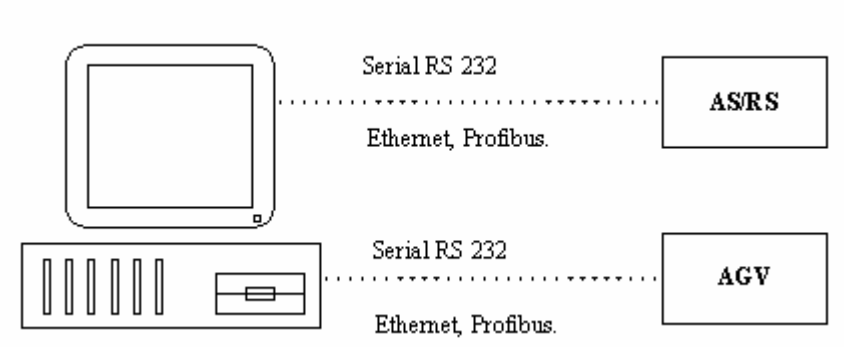


Figure 2.2 Communications PC between ASRS and AGV.

To establish the communication protocol we could do it through PROFIBUS DP, since this protocol is used in the stage of selection of pieces during the transport for the conveyor belt. However this idea was underrated due to the limitations of PROFIBUS DP with the access of two different maters to the same slave's information. An outline of the problem is given in the figure 2.3.

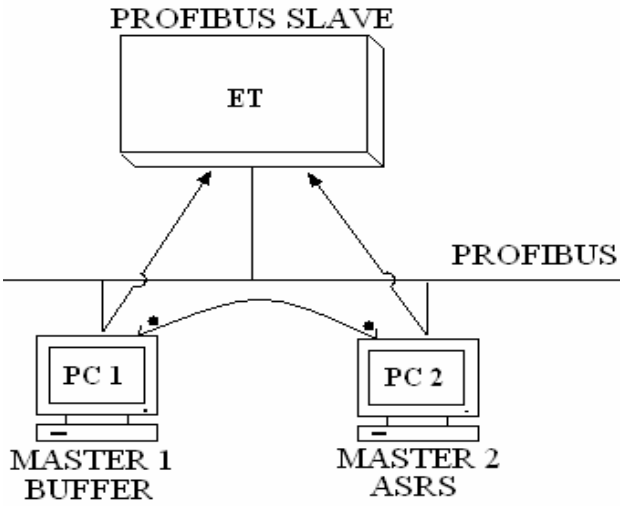


Figure 2.3 Limitation of PROFIBUS DP to access the same slave.

Due to we don't want to change the existing PROFIBUS network and as solution to this problem we has been opted to communicate the warehouse by means of the serial port and the control station of the AGV through the parallel port of the PC of our work station.

3. AUTOMATIC STORAGE AND RETRIEVAL SYSTEM (ASRS).

3.1 Working Principle.

The AS/RS is a storage and extraction system that it provides the placement and extraction of components in an automatic way (handling of inventories).

It works by means of Cartesian Coordinated, that which means that all their movements can be described in the XY plane. It has a robot that provides movement on the Z axis, as well as a "C" rotational axis relative to the movement XY; the C axis has a gripper, which is used to move away pallets of the warehouse or to put pallets inside the warehouse. The figure 3.1 shows two views of the warehouse.

Figure 3.1 Views of the warehouse.

The storage and automatic extraction system is composed by a series of components [5]:

- Robot Arm. It moves for warehouse according to the X Y Z coordinated.
- Gripper. It is the part of the mechanical arm that it catches the pallets of the warehouse. It is the robot's final effector and it is of pneumatic drive.
- Air pressure system. It provides movement to the gripper.
- Warehouse. They are the pigeonholes where the pallets are kept. There are 30 of these cells.
- DC Servo Drive. It converts AC in DC, which one works the warehouse.
- Keyboard and Monitor. It uses for giving orders and seeing it.
- Reading/Writing System. It uses to identify the pallets of the warehouse. It is integrated by a processor, an amplifier and a reader/writer head.
- The warehouse has the connection RS-232 DB-9 which allows an easy implementation with the serial port.

The ASRS control is carried out by means of software, which will be explained in the section 3.3.

3.2 Start up the System.

The first step is to check if the pressure air key in the right position, since otherwise the pneumatic system of movement didn't perform. Next we close the powered key. So the system is already for working.

We access to the operating system MS-DOS, where the name of the directory is typed (cd asrs) and its executable file (asrs). So we enter in the menu. The first image of the software, when we access to the menu, is given in the figure 3.2.

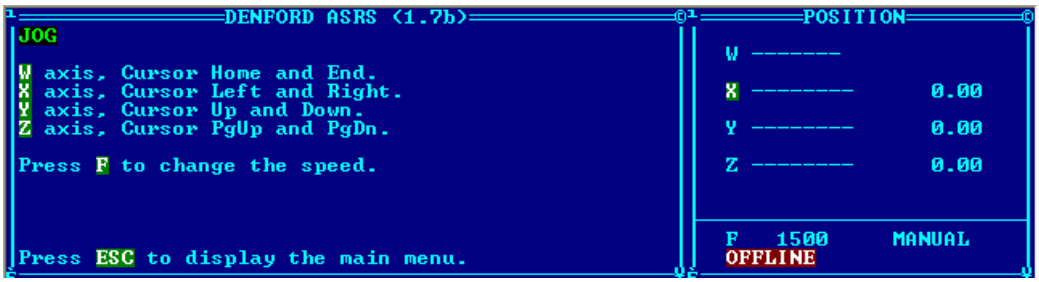


Figure 3.2 Initial image of ASRS software.

3.3 ASRS Control Application.

The ASRS software is used for controlling the warehouse and it is composed by several options. So, the options are:

- Home
- Jog
- Remote
- Local
- Settings
- Test
- Quit

3.3.1 Home.

To start the AS/RS it is necessary to take the robot arm to its mechanical zero. To access to the menu, we press the ESC key. Subsequently we access to the software through which the robot is driven. To take the warehouse to its mechanical zero we go to the option HOME of the menu and we pressed the keys of the keyboard X, Y, Z in this order.

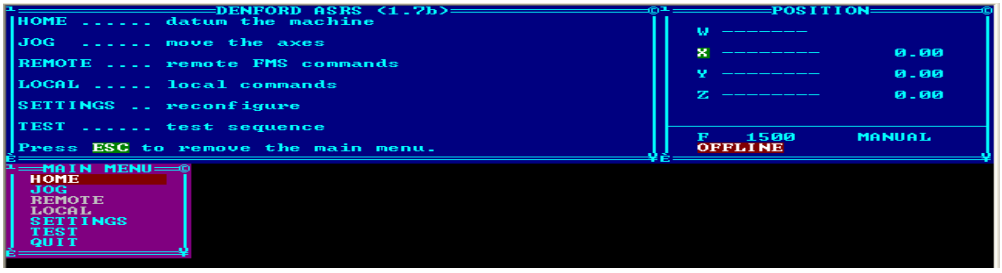


Figure 3.3 Main menu of ASRS control application.

For doing this operation it is very important to assure that the robot is in a safe position, free of any collision. We look if the gripper is outside of the cell and in closed position to avoid a crash when the mechanical arm moves in the X axis.

At this time the robot is ready to operate. We have several options to do the process .In a jog mode, in local mode and remote mode.

3.3.2 Jog Mode.

By means of manual mode (JOG in the menu) the robot arm is controlled through the keyboard of the ASRS in the following way [4]:

- With the cursors ← → the X axis is controlled.
- With the cursors ↑ ↓ the Y axis is controlled.
- With the keys *Page Up* and *Page Down* the Z axis is controlled.
- Lastly with the keys *Home* and *End* it is opened and closed the Gripper.

3.3.3 Local Mode.

In this option the robot arm is moved by means of commands that we must introduce. The used commands will be the following ones [4]:

- **MOVE *X***
- **FROM *X* TO *Y***

Here the letters "*X*" and "*Y*" are numbers of positions.

- *COMMAND "MOVE"*

With the command MOVE the arm moves directly to the position specified behind the command.

MOVE 1 → the arm goes to the position 1 of the warehouse defined previously in the program. Subsequently, we will explain how we can define positions.

MOVE 0 1 1 1→ the arm goes to the warehouse (ASRS) and it is located in the cell 11

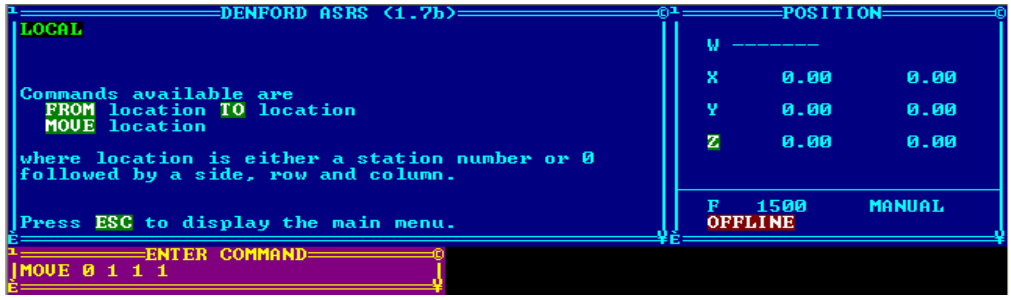


Figure 3.4 Instruction MOVE 0 1 1 1 in Local Mode.

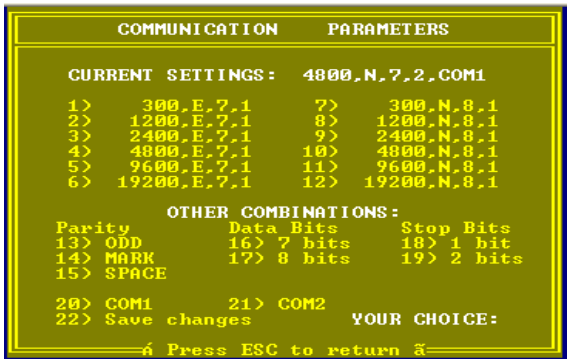
- *COMMAND "FROM... TO..."*

With the command FROM... TO... the arm moves from the specified position behind FROM to the position specified behind TO. The arm executes a sequence to load from the position behind FROM and a sequence to unload from the position behind TO.

For example: FROM 0 1 1 2 TO 0 1 1 1, it loads the pallet of the cell 1 2. Then the robot arm unloads the pallet in cell 1 1. By means of the key "F" the speed of the warehouse is controlled.

Note 1: The first two characters of the sequence, 0 1, define to the automated warehouse.

3.3.4 Remote Mode.



In remote mode, the commands are the same (MOVE and FROM … TO…), however the ASRS is controlled out of the station (via modem, hyper terminal…).

Figure 3.5 Configuration parameters to communicate ASRS.

For the control of the warehouse through the serial port, it will be necessary to establish the communication parameters between the warehouse and the port (bauds, parity or non parity, number of data bits, number of stop bit and the port used). So, in this operation mode we establish the communication parameters through the directory "PROCOMM". An image of the possible configurations of this directory is given above in the figure 3.5.

3.3.5 Settings.

In the Setting menu are several options. These options are SETUP, FIX PIGEONS, LOAD SETTINGS and SAVE SETTING.

When we access to SETUP option we find the next positions defined and amendable to the user's requirement: AS/RS, Code Tag, AGV, Robot 1 and Robot 2. In each one of these positions we will find the next possibilities: UNLOAD, LOAD AND POSITION.

Figure 3.6 Position defined in Setup option.

- UNLOAD and LOAD: These positions define relative movements to the position that POSITION indicates. The relative movements

are executed with the instruction FROM… TO: FROM (LOAD) TO (UNLOAD). The number of positions that you want to do it should be defined in the file ASRS.OPT.

- POSITION: It makes reference to the coordinates of the point which the mechanical arm arrives.

Others options inside the Setting menu are LOAD and SAVE SETTINGS:

- LOAD SETTINGS: It loads the existent configuration in the file ASRS.OPT.
- SAVE SETTINGS: It keeps the changes made in the file ASRS.OPT.

3.3.6 Test and Quit.

The option "Test" makes a sequence of movements for all the cells of the warehouse while the option "Quit" return to MS-DOS operative system.

3.4 Robot Collision.

In case the robot collision takes place the next process it should be followed:

- 1 To turn off the power supply, and to open the panel of controls.
- 2 To detect fuse broken.
- 3 To change the fuse.
- 4 To close the panel of control and to feed again the system at the same time we press the button AXIS LIMIT SWITCH OVERRIDE.
- 5 To take the robot to a sure position through the JOG MODE.
- 6 To take the robot to the initial position.

3.5 Defining a Position in the Warehouse.

All the positions of the warehouse are defined in the file ASRS.OPT [4], so if we want defined a new position, we must modified this file. It is convenient to explain the next. When we speak of defining a position, we refer to the positions which are in the submenu SETUP, inside the menu SETTINGS.

The first step is to access to the file ASRS.OPT, through the editor of texts. In this file we find defined all the parameters of operation of our ASRS. So the number of lines, number of columns, maximum speed, axes limit, cells coordinated, and sequence of movements to get a pallet are in the file. This parameter can be modified in this file.

An example to define a position is given in the annexes.

3.6 Turning off the warehouse.

To turn off the warehouse we should check that it is in a sure place, free of collision. Later, we select option “Quit” in the menu and then we cut the current.

4. AGV SYSTEM.

4.1 Introduction.

Automatics Guided Vehicles (AGVS) are designed to perform their operations without direct human guidance. They are used in a wide variety of industrial applications and can be laser, inertial or Cartesian-guided. A guided vehicle system, (or AGVS) consists of one or more computer-controlled wheel based load carriers (normally battery-powered) that runs on the plant floor (or if outdoors on a paved area) without the need for an onboard operator or driver. AGVs have defined paths or areas within which or over which they can navigate. A picture of our AGV system is given in the figure 4.1.[6]

Figure 4.1 View of AGV.

4.2 AGV Functionality.

AGV use this method for guidance: Fixed path, where a physical guide path (e.g., wire, tape, paint) on the floor is used for guidance. The AGV has two batteries. One of them is necessary for the movement for the floor. The other one is used to feed sensor and IR actuator that the AGV has. The AGV is controlled by means of control stations.

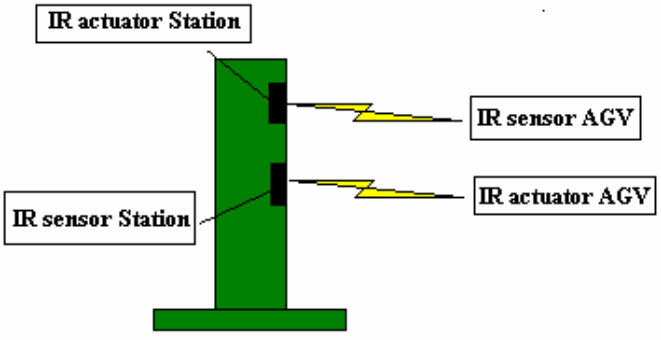


Figure 4.2 Control of AGV Station

The station is constituted also by one IR sensor and one IR actuator placed as it is shown in the figure 4.2. The station interacts with the AGV in the next way:

- The IR actuator of the station orders to stop as well as to restart the movement to the AGV.
- The IR sensor of the station indicates the presence or not of the AGV.

The control of the AGV will be carried out by the parallel port, as we have already commented**.**

5. SERIAL PORT AND PARALLEL PORT.

5.1 Serial Port.

5.1.1 Introduction to Serial Communication.

The serial communications are used to send data through long distances. There are two kinds of serial communications: synchronous or asynchronous. In a synchronous transmission the data are sent in blocks, the transmitter and the receiver are synchronized by one or more specials characters called characters sync.

The serial port of the PC is an asynchronous device. In an asynchronous transmission, a bit identifies the beginning and 1 or 2 bits identify the end, it is not necessary any synchronous character [7].

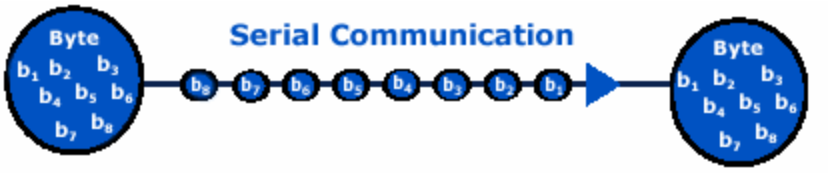


Figure 5.1 Layout of Serial Communication.

The serial port of a computer is an asynchronous adapter used to be able to intercommunicate several devices to each other. In our case we are going to communicate our work station (PC) and the warehouse. So, we will be able to control the ASRS from the mentioned work station.

An image of the serial port DB-9 male of our work station and a description of the pins is given in the annexes.

5.1.2 Data Transmission.

A serial port receives and sends information outside of the computer by means of a communication software or driver of the port series. We need to send a characters string to the warehouse, acquaintances FROM TO or MOVE.

So, a program is developed that sends the information (character string) to the port character to character, so the information is converted in a sign which can be sent for a cable series or a modem.

The transmission series begins with an initial bit; next the data bits are transmitted. Finally the bit stop is sent that indicate the end of the transmission of a character. The protocol allows use 1, 1.5 and 2 bits stop. For transmitting normal data ASCII the number of data bits used it is 7.

In the next figure 5.2 we can see the transmission of the character ASCII “A” that it is the 65, 01000001 in binary representation [8].

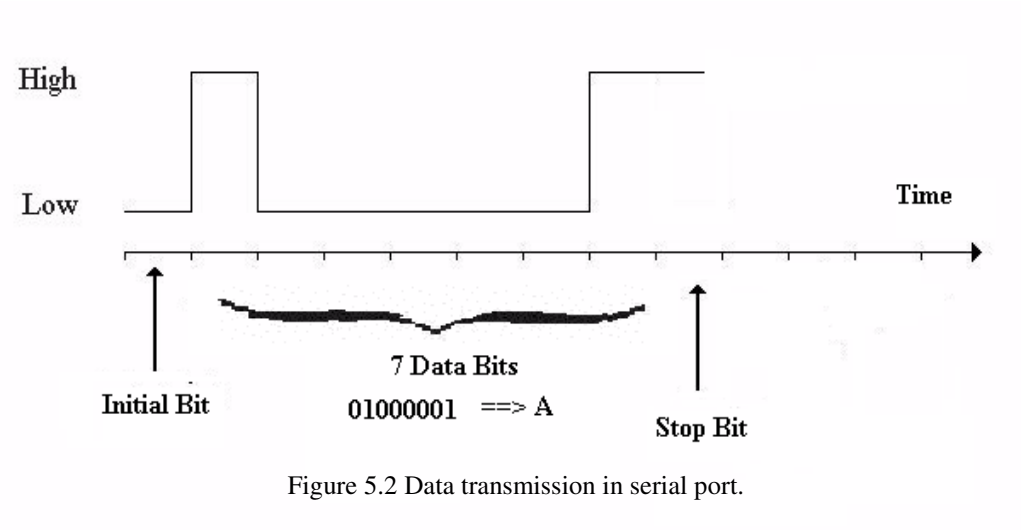


Figure 5.2 Data transmission in serial port.

5.1.3 Serial Communication Driver.

Our application is a program in assembler language which addresses and writes in the port. There are two modes of addressing the serial port, through the 14H interruption of the BIOS (we will use this) and through the 21H interruption of DOS. The 14H interruption of the BIOS use four functions to program the serial port. Each function is selected assigning a value to the register AH of the microprocessor.

The chart 5.1 shows the values that it should contain the registration AH to call each one of the services. The register DX should contain the number of the port series; COM1 is specified as 00h.

SERVICE	DESCRIPTION
00H	To initialize serial port
01H	To send data
02H	To get data
03H	To get the port state

Chart 5.1

To configure the serial port, it is enough with using the service 00h of the interruption, giving to the register AL the value corresponding to the parameters of the charts given in the annexes. With this method is possible to obtain transmission frequencies in a range from the 110 until the 9600 bauds.

For sending data for the serial port, it will be enough with giving to the registration AH the value 01h (Interruption Send Data) and put in the register AL the data for sending. Next, we invoke the 14h interruption [9].

An example of how we can send one data is given in the annexes.

5.2 Parallel Port.

5.2.1 Introduction to the Parallel Communication.

The parallel port greatly increases transfer speeds by using an eight wire connector which transmits the eight bits in a byte of data simultaneously, thus sending an entire byte of data in the time it takes to send a single bit in a byte of data is supplemented by several other handshaking signals, each sent on its own wires, which ensure that data transfer takes place smoothly.

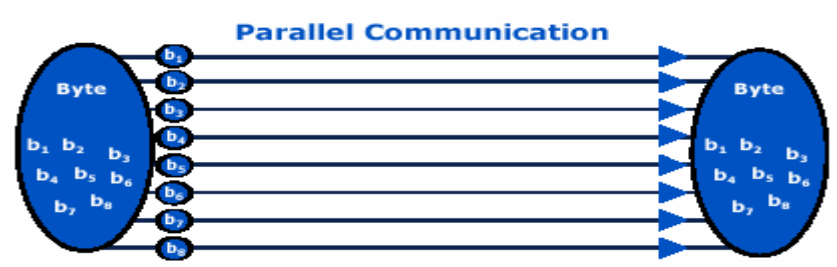


Figure 5.3 Layout of Parallel Communication.

The parallel interface provides the signals and hardware to transfer data one character at a time (8 bits in parallel) between the CPU and a device. The signals consist of 8 data lines and 9 handshaking (status and control) lines. An image of the parallel port DB-25 female of our work station and a description of the pins is given in the annexes.

The parallel port is supposed formed by three registrations:

- Data Register
- Status Register
- Control Register

The addresses available for the data, status, and control register of an adapter are shown in chart 5.2. BIOS determines (during the restart initialization) which addresses have printer adapters installed. Normally these values are standard.

	Data Port	Status Port	Control Port
LPT 1	03BC	03BD	03BE
LPT 2	0378	0379	037A
LPT 3	0278	0279	027A

Chart 5.2 [10]

5.2.2 Control AGV Station.

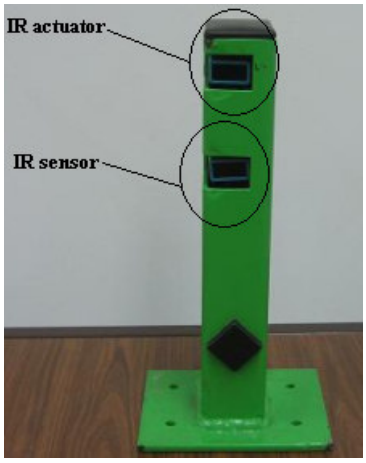


Figure 5.4 View of AGV Station.

In our case the parallel port will be used for the control of AGV station. So, we will develop a software application which writes and reads in parallel port.

This way we can know when the AGV is absent or not (IR sensor), and to order stop or restart the progress (IR actuator).

5.2.3 Parallel Communication Driver.

To work with the parallel port we need to know the base address assigned by the BIOS. We can use a called program Debug.exe that indicates us the assigned address in MS-DOS. Now, we can introduce the following commands.

It is enough with typing the word debug, the program responds placing a sign of less "– ", without spaces we type d040:08L8 and we press the key enter, then the program debug.exe indicates us in a series of numbers the address for the available parallel port in our system. When we obtain the information we close the program Debug.exe typing the letter "q" and pressing enter. Now, we get the parallel port address, so we can develop the software for writing and reading in the port [10].

The writing of the port is made in the data register while the reading is made by means of the state register. The writing of the port is necessary to order the AGV stop and restart the progress. The writing is made by means of the pin 4 of the port. The reading is necessary to know when the AGV is absent or not. The reading is made by means of the pin 10 of the port (one ==AGV is not absent; zero==AGV absent) [11].

5.2.4 Voltage Level of Parallel Port and Station.

PC parallel port can be damaged quite easily if you make mistakes in the circuits you connect to it. If the parallel port is integrated to the motherboard (like in many new computers) repairing damaged parallel port may be expensive (in many cases it is cheaper to replace the whole motherboard than repair that port).

The following problem to be solved is the different voltage levels of the port and of the station. Two sensors exist. The IR actuator, that orders to stop and to restart to the AGV, works with the next voltage level:

- Stop AGV == 8 to 28 V
- Go AGV == 0V

The IR sensor, that indicates the presence or not of the AGV, works with the level:

- AGV is not absent == 5 V
- AGV is absent == 0V

We already know the voltage level of the station. Now we must know the voltage of our parallel port. The sensor 1 will be controlled by means of the data register, while for sensor 2 we will use the state register.

The data register has a voltage level of 3.6 V. This register will be used to write in the port and order or not to the AGV stop (control sensor 1). Therefore we need to convert 3, 6 V in 8-28V. This voltage conversion is made by means of the following circuit:

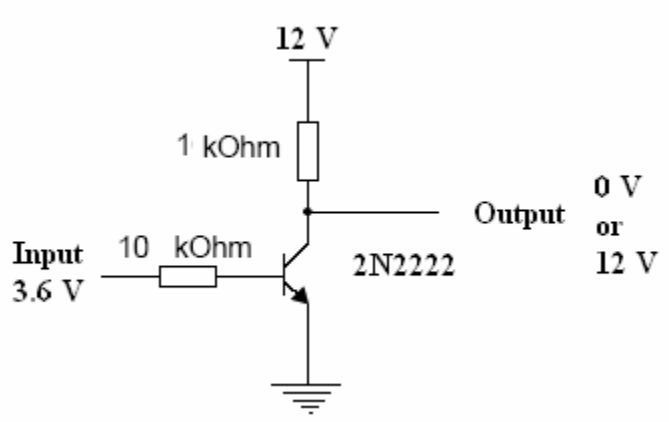


Figure 5.5 Circuit voltage converter.

The resistor values may have to be modified depending on the switching speed required. Therefore if we write a logical zero in the port we will obtain 12V, while if we write a logical one, we will have 0V. This way, we solve the problem of the voltage conversion.

The state register of the port will be used to read the absence or not of the AGV (control sensor 2). The voltage level of the register state is 4.6V. In this case, it would not be necessary any conversion, since the sensor 2 work to 5 volts (it is also necessary to keep in mind the voltage fall in the cables). In addition, the parallel port supports an over voltage of around 1V [11].

6. INTEGRATION WITH SCADA SYSTEM: WinCC.

6.1 Definition of SCADA System.

SCADA is an acronym for Supervisory Control and Data Acquisition. The SCADA system uses the computer and communication technologies to automate the industrial processes control. These systems are integral parts of most of the complex or very geographically dispersed industrial atmospheres since they can collect the information of a great quantity of sources very quickly, and they present it to an operator in a friendly way. The SCADA systems improve the effectiveness process and control providing the necessary information for making decisions.

6.2 SCADA Tool: WinCC.

6.2.1 Introduction.

WinCC is SCADA tool for developing HMI (Human Machine Interface). WinCC's user-friendly components permit the trouble-free integration of new or existing applications. With WinCC, you have a process visualization program that enables you to easily observe all aspects of your automation processes.

The WinCC Project Engineering Environment:

- Pictures - to design plant representations.
- Archiving - to store data/events, with a time stamp, in a SQL database.
- Report Designer - to generate reports for the requested data.
- Data Management - to define and collect data through out the plant.
- WinCC Runtime
- Enables machine operators, who are on the plant floor or in a control room, to interact with the machine application [1].

6.3 Overview of the SCADA Application.

By means of the use of the tools of WinCC we will develop an application through which the warehouse will be integrated in the global system. For the development of the application we have used the drivers created for the communication between the warehouse and the AGV with the PC since WinCC doesn't contain these drivers.

The visualization of the warehouse and of the AGV in the monitor of the work station has been carried out by means an image created in WinCC. In the same way, a database has been created in EXCEL which is communicated with the created image of the warehouse. In this database we introduce the kind of pieces which there are in the warehouse. These pieces are of three types: white (value 1 in databases), black (value 2 in databases) or gray (type 3 in databases). These pieces will appear drawn in the image of warehouse of our application. When the warehouse retires a piece to deposit it on the AGV, the picture of the piece will disappear of

the cell of the warehouse. We have also carried out a series of warnings reminders of the steps to give when we turn on the warehouse. We will explain the developed application as well as the tools used in WinCC.

6.4 Control Application.

6.4.1 Design the Process.

The first step is to design the process that the warehouse will carry out when it catches the pieces. We have designed two different processes that we are going to explain now. The processes get all the pieces of the same kind and the pieces are deposited on the AGV, but with a difference. While in the process one the AGV waits that robot arm catch all the pieces of the same type for going again, in the process two the AGV waits only that the robot arm catches one piece to renew its progress. The next flow diagrams represent with clarity the processes.

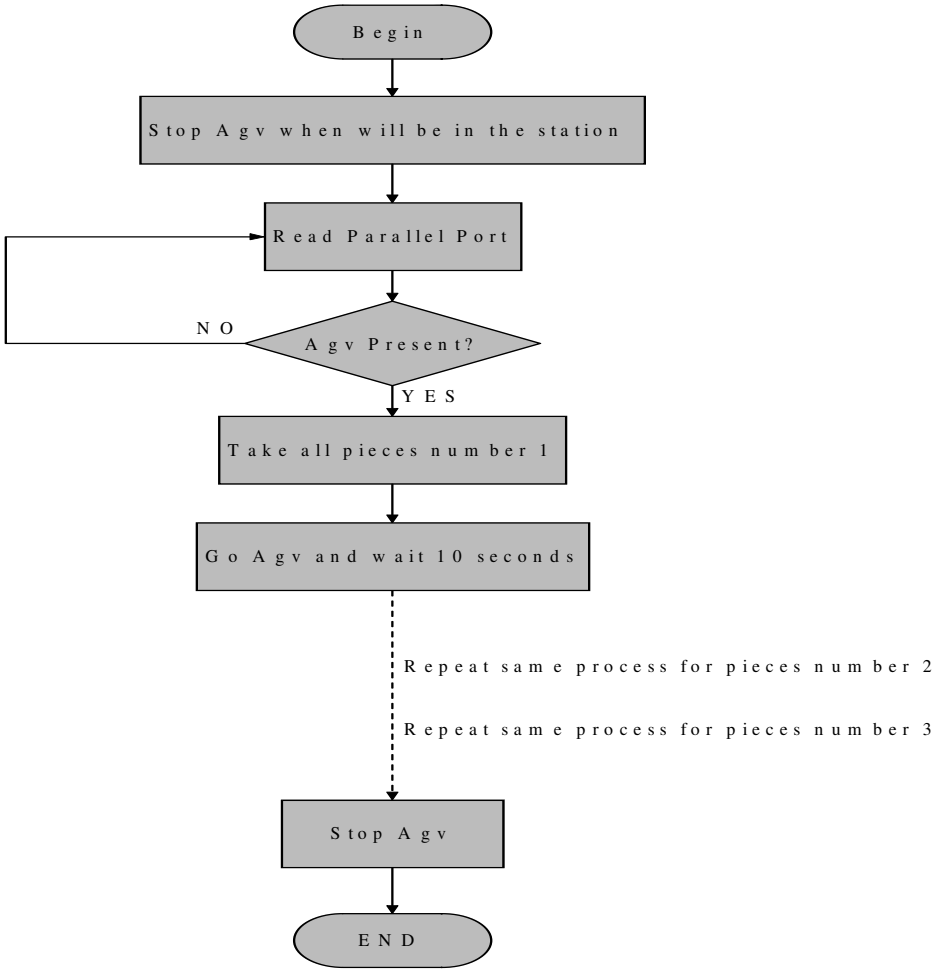


Figure 6.1 Process One.

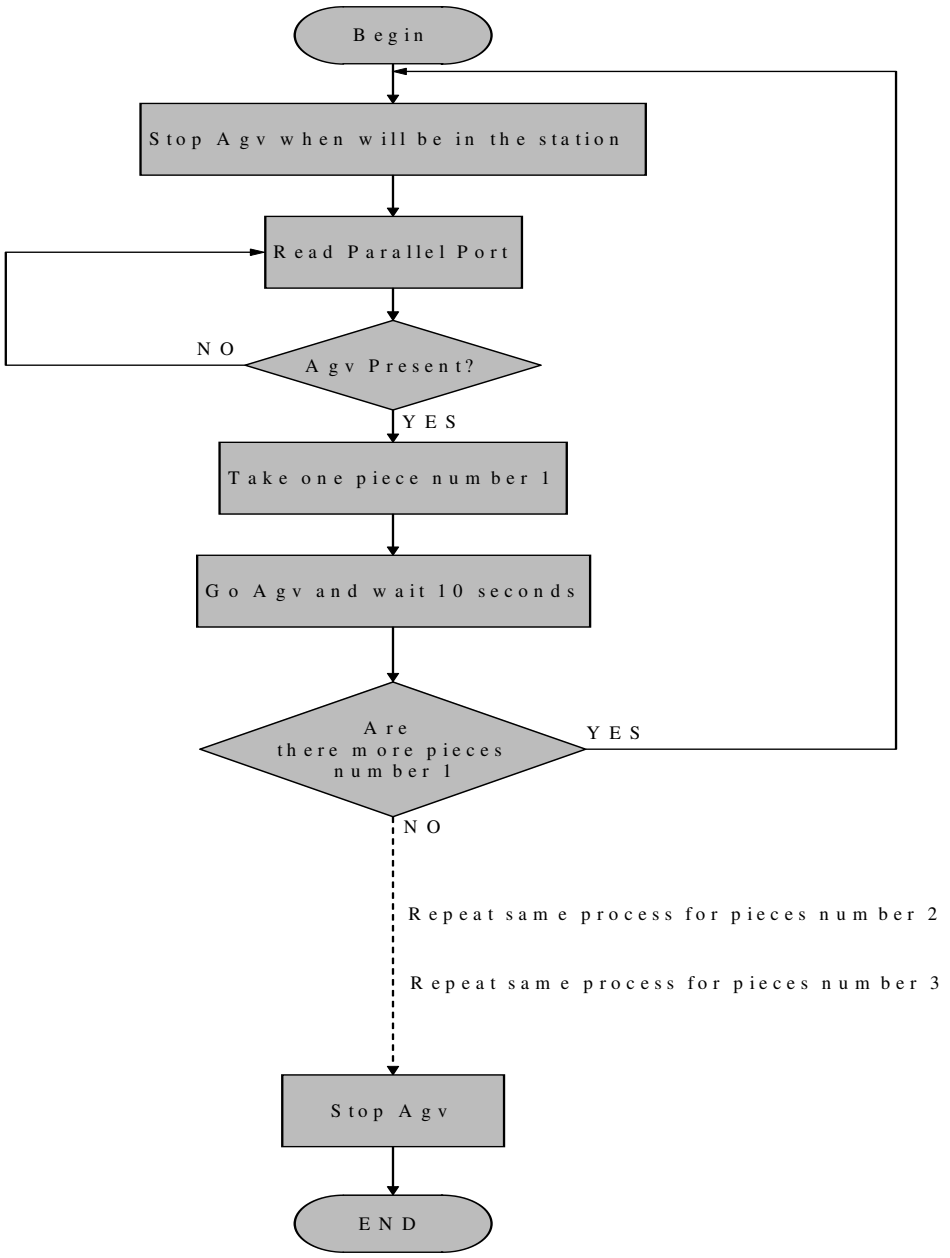


Figure 6.2 Process Two.

Designed the processes we can begin with the development of the application.

6.4.2 Creation of Variables.

The next step is to create the necessary variables for developing our processes. In WinCC we can create two types of variables or tags: internal variables and process variable.

Tags with values supplied by the process are referred to as process or external tags in WinCC. In the case of process tags it is necessary the addition of a driver to create it. The associated tags are created in the directory structure of this communication driver [1].

For the communication between our application and the database it is necessary the creation of 30 process variables (one of them for each cell). The database is already created in EXCEL.

The communication between the database and the application is made by means of the addition of a driver, in this case WINDOWS.DDE. When we have established this driver, we make a connection between the database and the application. Now we can create the necessary process variables. Each one of the variables is associated and represents a cell of the warehouse, as well as a cell inside the Excel database. The value of these variables in the database will be reflected in the warehouse of our application with the image of different pieces.

A scheme of the process to communicate the database and the warehouse of our application is given in the figure 6.1.

Figure 6.3 Scheme to communicate database and warehouse.

In the figure 6.2, we show the driver, the connection and the process tags in WinCC.

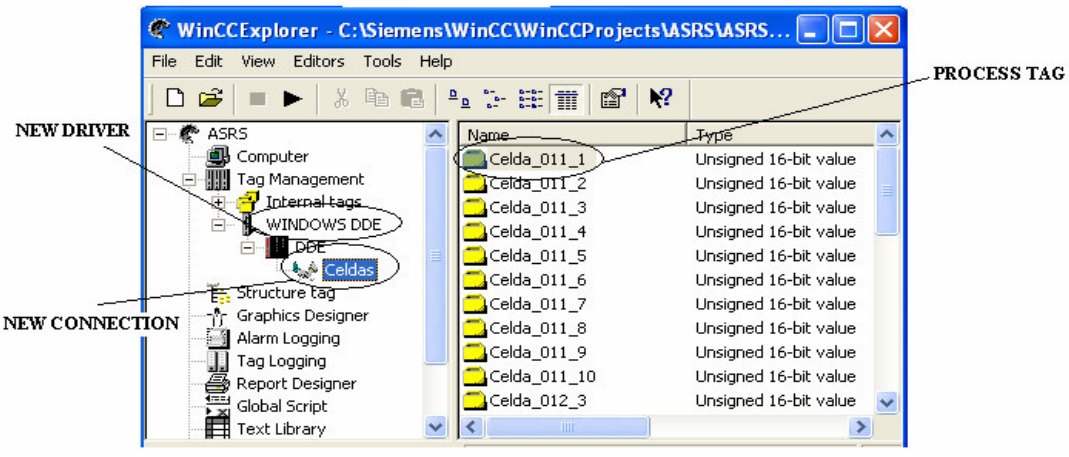


Figure 6.4 WinCC windows with the diver, connection and processes tags.

Tags not supplied with values by the process, known as the internal tags. In our application we have been created two internal variables called “AGV “and “Test”. According to the value of the variable “AGV” different pictures of the AGV are shown (absent or not). We will explain later on how make this inside the Graphic Designer (WinCC Tool). The variable “Test” is used in the program code to read the parallel port. According to the value that takes (1 or 0) it is continued reading or not the port.

6.4.3 Using Created Drivers.

Declared the necessary variables, we develop the programs which will make the two processes. WinCC has a C language compiler called Global Script, with some characteristics. One of them is the possibility to execute any program installed in the PC by means of the instruction PROGRAMEXECUTE (path of the file). By means of this instruction we execute the programs developed to send information to the ports. An example would be:

ProgramExecute("C:\\Siemens\\WinCC\\WinCCProjects\\ASRS\\Programs\\PORTZERO .EXE"). Being PORTZERO.EXE the executable file of the program that writes a logical zero in the parallel port. In this compiler, there are standard functions that execute diverse actions as for example “Exit Runtime”, “Copy Tag Valued”, “Exit WinCC or Windows”...[1]

6.4.4 Using the Graphics Designer.

Finished the developing of our processes, the following step is the creation of images which represents them. In this chapter we will create the image of the warehouse and of the AGV. For this; we will use the tool Graphics Designer of WinCC.

During configuration, the Graphics System is used to create the pictures which display the process in runtime. The Graphics Designer is an editor for creating process pictures and making them dynamic.

6.4.5 Creation of Images.

For the creation of images we will use of the graphic editor, Graphic Designer, where we will define our animations which will be controlled by internal and external variables that we have been created previously. In the Graphics Designer the predefined graphic elements which enable efficient creation of process pictures are called "objects". All objects can be easily inserted into a picture from the object palette. The list of the objects is given in the annexes[1].

Firstly, we create the image of the ASRS and the AGV. With the help of a digital camera, we take pictures of the warehouse and the AGV. These pictures will be inserted to our images editor.

To create the warehouse and the AGV, we have used the tool STATUS DISPLAY of Graphics Designer. With this tool, we can associate to each variable one image. When a change

takes place in the variable a change takes place in the image. This way, to the internal variable "AGV" is associated to the image of the AGV. If the variable "AGV" is one, the AGV is shown. If the variable "AGV" is to zero, the AGV is not shown.

The process variables are associated to the cells pictures of the warehouse. Depending on the value that we give to the cell in our database the image in the warehouse will vary. The possible values are:

- Zero →Pallet Empties
- One →White Piece
- Two →Black Piece
- Three →Gray Piece

A figure of the warehouse with several pieces in the Graphics Designer is given above.

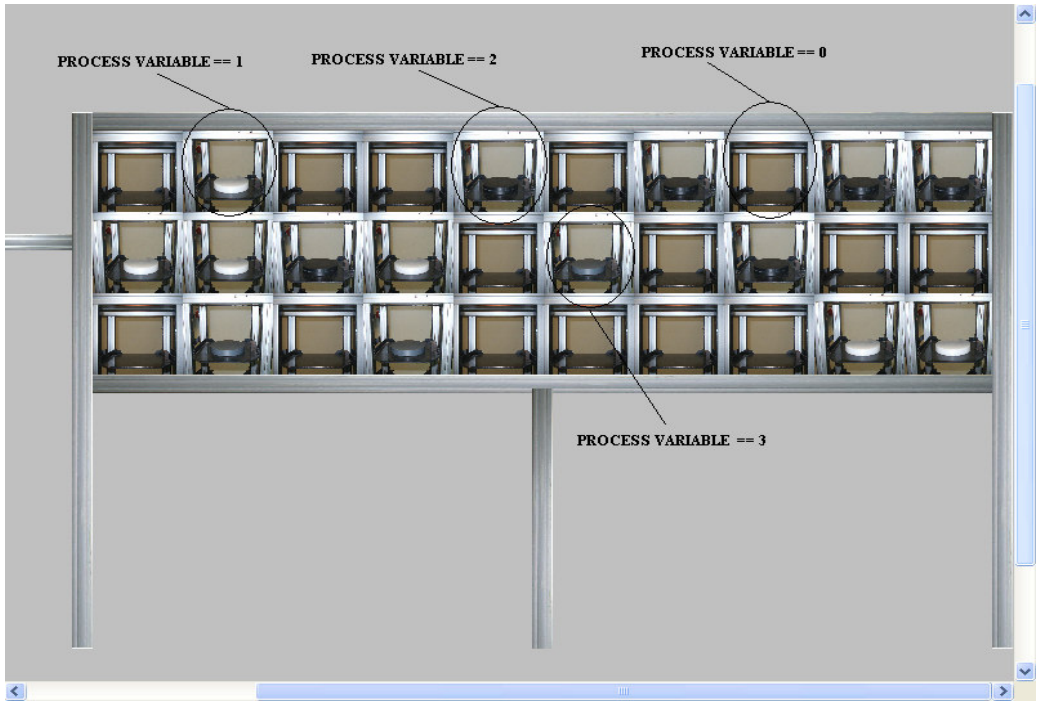


Figure 6.5 WinCC windows of warehouse with several pieces and processes tags values.

6.4.6 Execution of the Process.

Created the main image of our application, the next step is the execution of the different processes developed to catch pieces of the warehouse.

The first step will be to create two buttons, through the graphics editor. One of the buttons will represent the process one and the other one will represent the process two.

The following step is assign to each button the execution of each process developed in the compiler of WinCC. This way, when we press one of the buttons the process one or the process two will be started. This process is designated in WinCC "EVENT" inside the properties of each object.

Figure 6.6 Buttons of Process One and Process Two.

Lastly, we make a button which has assigned the function to stop the execution of the processes.

6.4.7 Manual Mode.

The application is able to develop the execution of two different processes. It is obvious that we can design as many processes as we want**.** Due to this we have considered convenient the creation of a different system which we have called Manual Mode. In this system a user decides that pieces should be chosen of the warehouse to put them on the AGV.

So in the developed image of the ASRS, we have been created in each cell two bottoms. One of them (the red one) executes the operation of taking the palette to the AGV (send the instruction FROM... TO ...to serial port). The other one (the blue) it executes the instruction MOVE, commented in the chapter 3.4.3.

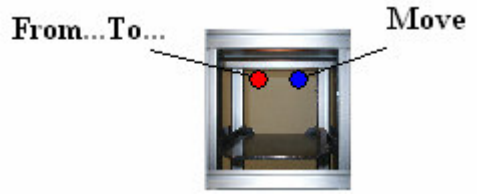


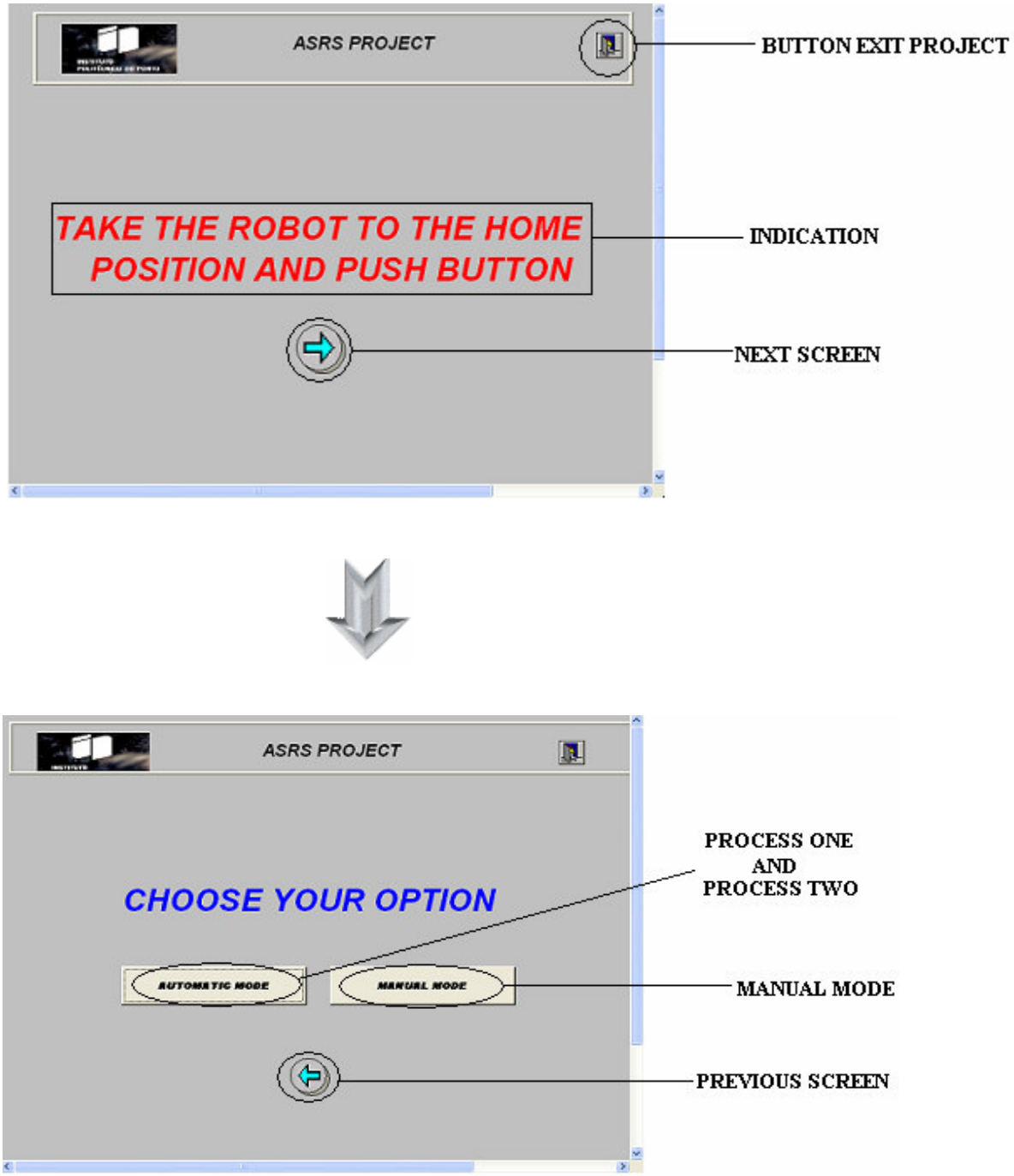
Figure 6.7 Cell of ASRS in manual mode

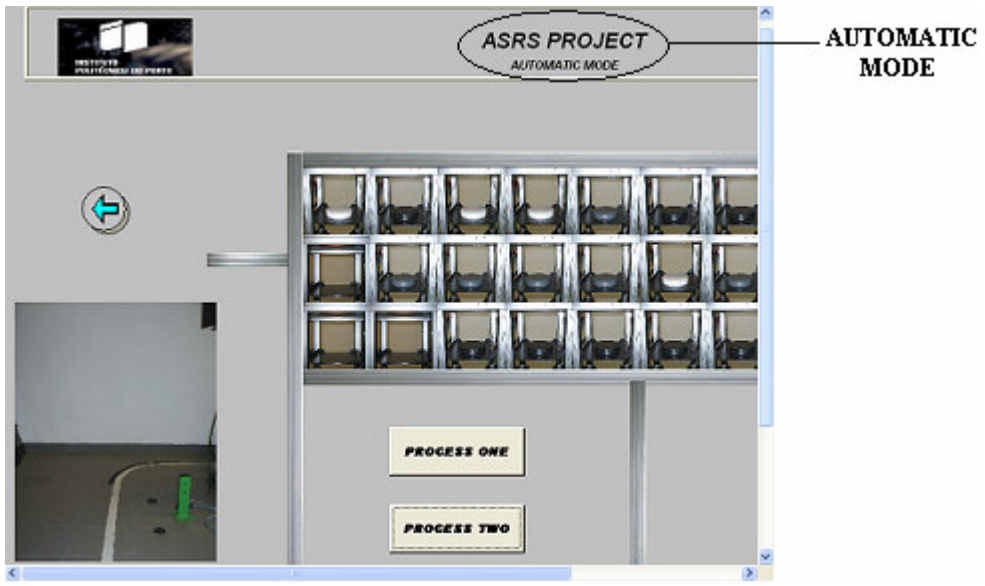
Two buttons which order the AGV to stop and to restart the progress also exist. These buttons will also be controlled by the user.

6.4.8 Concluding the Application.

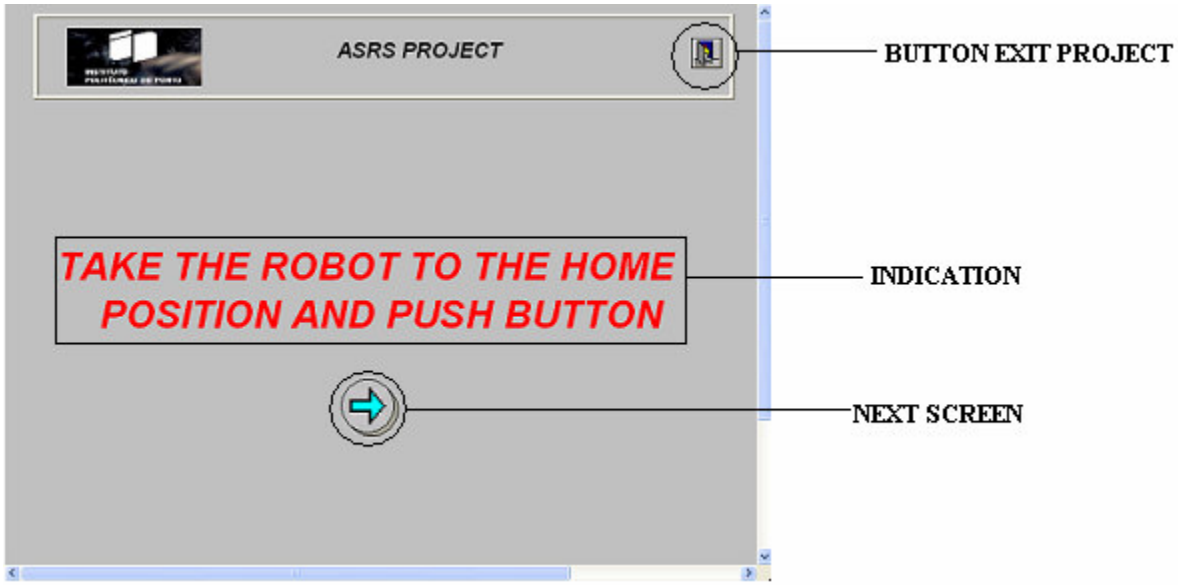
Finished the main application, it is convenient the creation of some initial screens. This will be the beginning of the application and it will remind us the steps to give when turning on the warehouse. Through these screens we guide the user which can choose the wished option, the processes one and two (called automatic mode) or the manual mode. An outline of the final process is given in the figure of below.

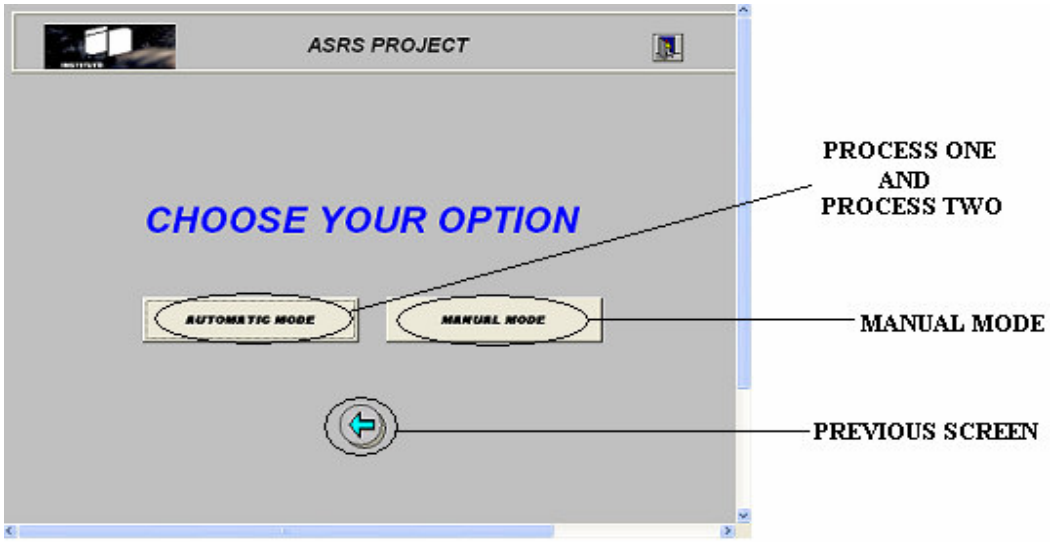
Option A:

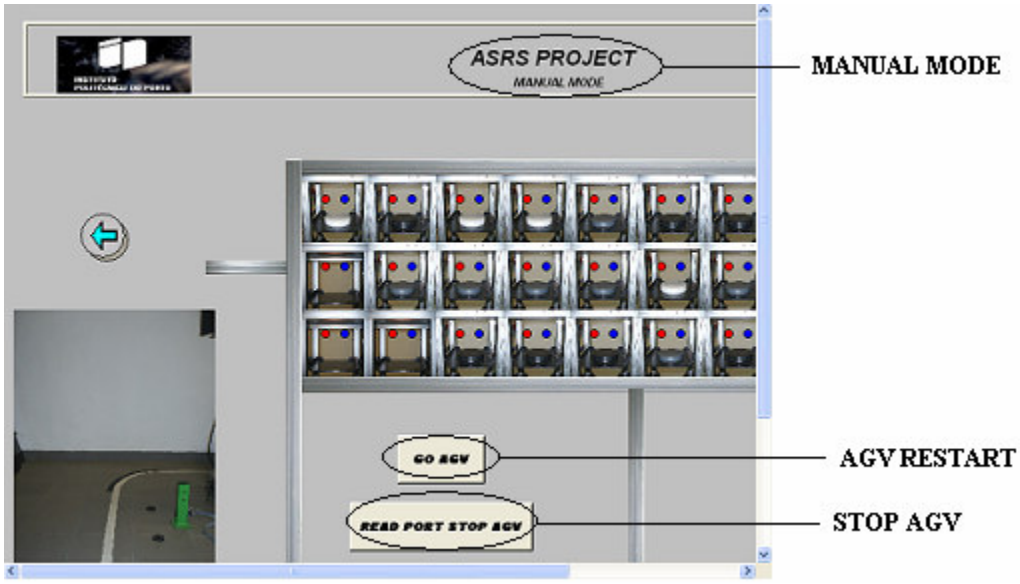



ASRS PROJECT
AUTOMATIC MODE
AUTOMATIC MODE
PROCESS ONE
PROCESS TWO

Option B:


ASRS PROJECT
BUTTON EXIT PROJECT
TAKE THE ROBOT TO THE HOME POSITION AND PUSH BUTTON
INDICATION
NEXT SCREEN


ASRS PROJECT
CHOOSE YOUR OPTION
AUTOMATIC MODE
MANUAL MODE
PROCESS ONE
AND
PROCESS TWO
MANUAL MODE
PREVIOUS SCREEN


ASRS PROJECT
MANUAL MODE
MANUAL MODE
GO AGV
READ PORT STOP AGV
AGV RESTART
STOP AGV

7. CONCLUSIONS.

Described the characteristics of the application, we can conclude that the integration of the warehouse in the manufacturer system has been possible by means of the communication by serial port and WinCC. The connection of these two tools makes possible easy and effective integration of the automated warehouse AS/RS to any industrial process with success**.**

We can conclude also that the PROFIBUS DP limitations (to access two masters to the same slave) are resolved to communicate the warehouse through the serial port.

With concern to the parallel port we can conclude that it is also a very useful device for the communications, and that it is usually thought only for the printers. It is demonstrated that the port can be used for more uses.

During the realization of this project we have been several difficulties related with the operating system used in the work station, WINDOWS NT. It has been a significant effort to use this OS platform because below this operating system the connection DDE between the database of the system and WinCC is not possible, as well as the access to the parallel port is restricted, since it is not allowed to read. For this reasons, we make the decision of changing the operating system WINDOWS NT to WINDOWS XP.

Other aspect of the project is the reading/writing system built in the warehouse possesses. It has been tried to put into running this system but without success. We have contacted with the manufacturing company of the system (BALLUF) to look for solutions. We have tried opening the processor of the system for checking the configuration. Configured the processor was not successful.

8. BIBLIOGRAPHY.

8.1 Books.

- Simatic HMI WinCC Version 5.
  User´s Manual.[1]
- Programación en Lenguaje Ensamblador
  Antonio Moreno Fernández-Caparrós [2].

8.2 Webs.

1. ASRS and AGV.

- http://www.hurray.isep.ipp.pt/rfpilot/ [3]

- http://atlas.puj.edu.co/ftp/centros/cap/manual_asrs.pdf [4]
- http://www.chi.itesm.mx/~cim/practicas/pract1.html?componentes [5]
- http://www.globalspec.com/LearnMore/Material_Handling_Packaging_Equipment/Automatic_Guided_Vehicles_AGV [6]

2. SERIAL PORT AND PARALLEL PORT.

- http://www.ctv.es/pckits/tpseriee.html [7]
- http://www.ipn.mx/sitios_interes/sanlovdra/serie.htm [8]
- http://members.tripod.com/hgr/puertos_ensamblador.html [9]
- http://www.modelo.edu.mx/univ/virtech/circuito/paralelo.htm[10]
- http://www.hut.fi/Misc/Electronics/circuits/parallel_output.html [11]

3. SCADA WINCC

- http://www.esi2.us.es/~gordillo/labcpc/ScadaWinCC.pdf
- http://www.ad.siemens.com.cn/download/manual/docu_pdf/as/hmi/Manual/WinCC%20Basic%20V6.pdf

-ANNEXES-

ANNEXE 1

EXAMPLE TO DEFINE NEW POSITION IN WAREHOUSE

The positions are defined by the letter "S", so "S 4" means that there are four defined positions. Next we find the name of the positions:

S_1 ASRS
S_2 Code tag
S_3 Robot1
S_4 Robot2

Later on we define the Cartesian coordinates of each position, as well as the numbers of relative movements to this position.

Example:

We change S_4 for S_5 (new position)
S_5 New Position
S_5_X 2000.00
S_5_Y 60.00
S_5_Z 500.00
S_5_OFF 2
S_5_ON 1

Now we define the Cartesians coordinated of the relative movements.

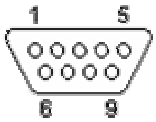
S_5_OFF_1_X 200.00
S_5_OFF_1_Y 0.00
S_5_OFF_1_Z 0.00
S_5_OFF_1_D 5000.0
S_5_OFF_1_W 0
S_5_OFF_2_X 200.00
S_5_OFF_2_Y 125.00
S_5_OFF_2_Z 0.00
S_5_OFF_2_D 5000.0
S_5_OFF_2_W 0
S_5_ON_1_X 0.00
S_5_ON_1_Y 0.00
S_5_ON_1_Z 0.00
S_5_ON_1_W 1
S_5_ON_1_D 0.0

Depending on the value of "W "the gripper is open or closed (1 or 0). The letter "D" means the speed of the mechanical arm. Lastly, we save the changes and we quit the editor of texts.

In the next figure, we show the new position in the submenu SETUP.

ANNEXE 2

IMAGE AND DESCRITION OF PINS OF SERIAL PORT.



PIN	PURPOSE	SIGNAL NAME
Pin 1	Data Carrier Detect	DCD
Pin 2	Received Data	RxData
Pin 3	Transmitted Data	TxData
Pin 4	Data Terminal Ready	DTR
Pin 5	Signal Ground	Gnd
Pin 6	Data Set Ready	DSR
Pin 7	Request To Send	RTS
Pin 8	Clear To Send	CTS
Pin 9	Ring Indicator	RI

ANNEXE 3

PARAMETERS OF THE REGISTER "AL" TO CONFIGURE THE SERIAL PORT

BITS								
7	6	5	4	3	2	1	0	**USE**
X	X	X	.	.	.	.	.	BAUDS
.	.	.	X	X	.	.	.	PARITY
.	.	.	.	.	X	.		STOPS BITS
.	.	.	.	.	.	X	X	DATA BIT

BAUDS	
BITS	**SPEED**
000	110
001	150
010	300
011	600
100	1200

PARITY	
BITS	**MEANING**
00	NON
01	ODD
10	NON
11	EVEN

STOP BIT	
BITS	**MEANING**
0	ONE
1	TWO

DATA BIT	
BITS	**MEANING**
00	NO USED
01	NO USED
10	7
11	8

ANNEXE 4

HOW SEND A DATA BY SERIAL PORT IN ASSEMBLER.

Our configuration is:

- 4,800 bauds
- Non parity
- 2 bits stops
- 7 data bits

So, we must do:

```
MOV AH, 0H      ; Initialize serial port
MOV AL, 0C6H    ; Our configuration
MOV DX, 00H     ; Port COM 1
INT 14H         ; Interruption
```

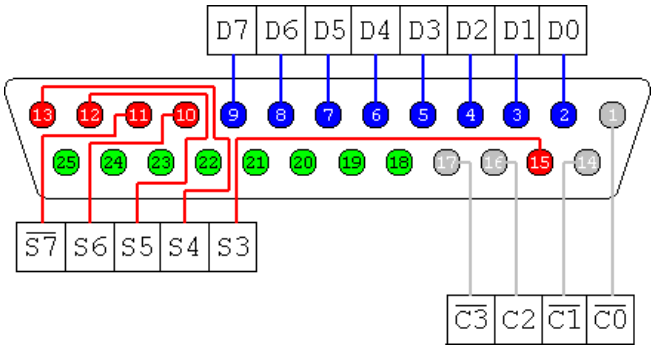
Now the serial port is configured. Let send one data:

```
MOV AH, 01H     ; Send data
MOV AL, "A"     ; Data sent
INT 14H         ; Interruption
```

So, we can send character string form our work station to our warehouse. In our application the instruction LODSB has been used, which permits to take consecutively to the register AL each one of the letters of a characters string previously defined [2].

ANNEXE 5

IMAGE AND DESCRITION OF PINS OF PARALLEL PORT



PIN	PURPOSE
Pins 2-9	Data Bits
Pin 18-25	Ground
Pin 10	Acknowledge
Pin 11	Busy
Pin 12	Out of paper
Pin 13	Selected
Pin 15	Error
Pin 1	Strobe
Pin 14	Autofeed
Pin 16	Initialize
Pin 17	Select

ANNEXE 6

OBJECT LIST OF GRAPHICS DESIGNER

Standard Objects	Smart Objects	Windows Objects
Line	Application Window	Button
Polygon	Picture Window	Check Box
Polyline	Control	Option Group
Ellipse	Ole Element	Round Button
Circle	Both	Slider
Ellipse Segment	Bar	
Pie Segment	Graphic Object	
Ellipse Arc	Status Display	
Circular Arc	Text List	
Rectangle	3-D Bar	
Rounded Rectangle	Group Display	
Static Text		
Connector		