The System

In the previous chapters which introduced the programmable logic controller, a discussion on CPU, P/M, I/O modules, the peripherals and their use in enhancing the PLC was evolved. This chapter looks inside the PLC and investigates its internal operation by actually designing a PLC hardware system.

The total system hardware is designed according to the available facilities from the Shivaji University Electronic Department Laboratory and assistance provided by Prof. A.R Nigavekar (KIT's College of Engineering Kolhapur). It was built and tested in the laboratory successfully and submitted for evaluation to Prof. A.R.Nigavekar.

PLC: A Look Inside

This deals with the investigation of the four main elements, while analyze the make up of discrete input and output modules(interfaces), by seeing how they can be interfaced. It examines what goes inside the PLC to enhance the understanding of the remaining topics to come.

- The total system hardware is divided into four major cards and the power supply.

The division is being done on the basis of the function of the individual cards.

- CPU Card
- System Card
- Input Card
- Output Card

The CPU Card: Central Processing Unit

This card is designed, using Z80 microprocessor around which the whole system is built. This card also houses the entire memory decoding and data bus drivers. A four MHz system crystal oscillator is used as the clock source for Z80 CPU.

An octal bus transceiver 74LS245 is used as the driver for data bus (D0-D7). The line drivers 74LS245 are also used to drive the address bus (A0-A15) and control line signal.

The memory system housed in this case has IC 2732 which contains the monitor program for the system with address from 0000 to 0FFF. IC 6116 is used as scratch pad memory with address from 1000 to 17FF. IC 2764 which contains the ladder program has the address from 6000 to 6FFF. (REF 35,36,37)

The decoding system on this card incorporates two 4 to 16 decoders and a 2 to 4 decoder. The line A0 to A3 gives 16 output lines through 74LS154 (4 to 16 decoder). These lines are used as chip select lines. The next 4 address lines (A4 to A7) are used as card select lines.

The address lines A0 to A15 are given to 74LS224 drivers, to drive 4 slot select lines for selecting the system slot. The Z80 is used for addressing as the CPU has as unique feature in which the MSB address lines i.e. A0 to A15 are duplicated with the contents of the register B. These facilities increased input and output capability of the system.

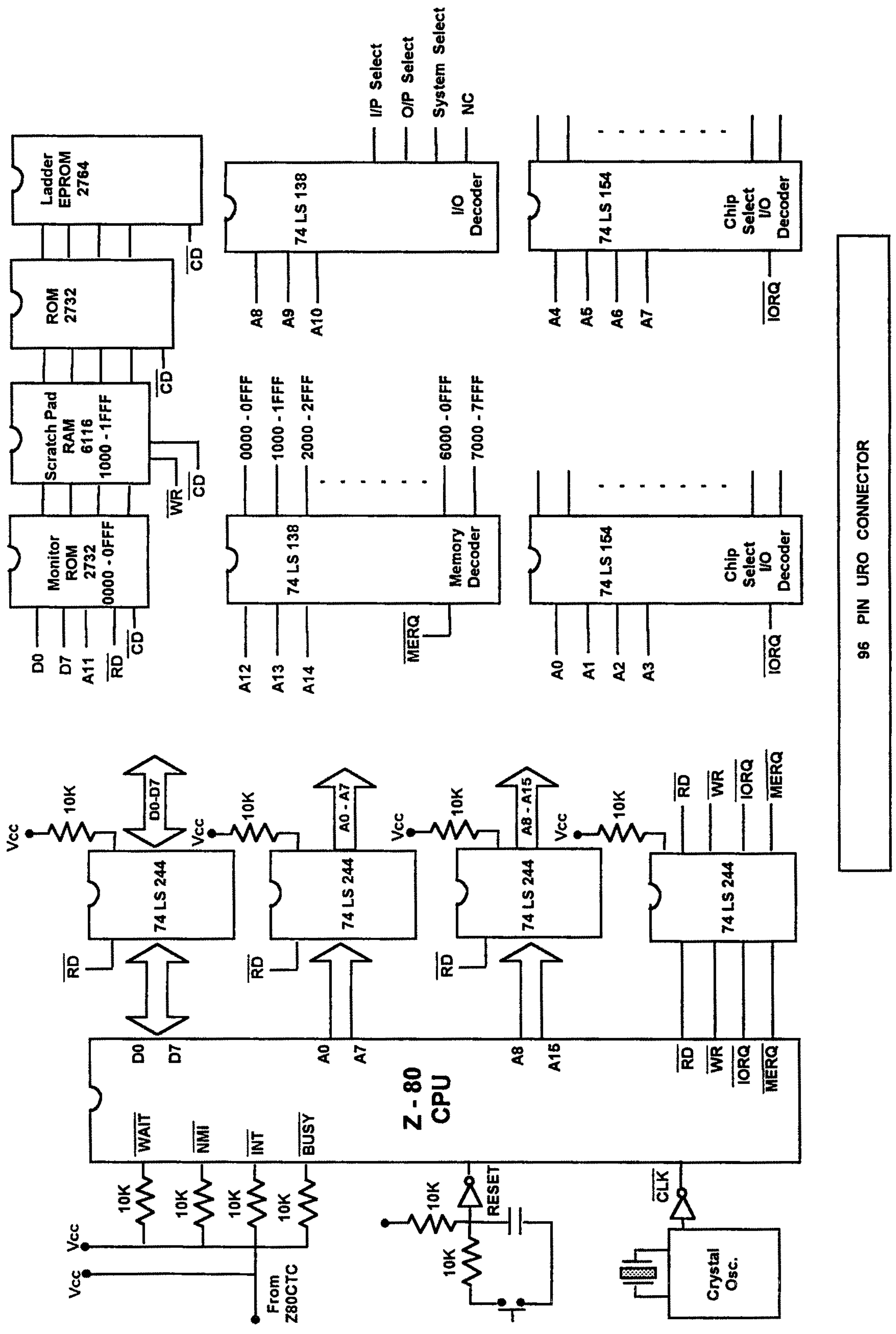


Figure 9 CPU Card

The System Card: Supporting Unit

This card incorporates the following main ICs in the system card, IC 8255, IC 8279 IC Z80CTC.

These ICs are selected by the 3 address lines A2-A4 using 74LS138 decoder, which generate the chip select signals.

The IC 8279 is used to control the keyboard containing 28 keys and seven segment display. This interface was required during the initial stage of software development (low level format), while creating the ladder control software, this makes works easily. It is needed in case of some special functions.

The two IC 8255 are used for the interfacing of ADC's and DAC's useful in controlling devices like stepper motor, etc.

The Z80CTC is used as the timer, each chip houses 3 counters/timers. Three different time bases are provide by 3 IC 555 timers with time buses of 0.01sec, 0.1sec respectively. (REF 35,36,37)

Input Card: Input Interface

For the circuit to perform properly, the input signal to the PLC must be boosted to a power level compatible with the CPUs logic circuitry. The input card is used in the system through which the CPU reads the status of the external controlling element.

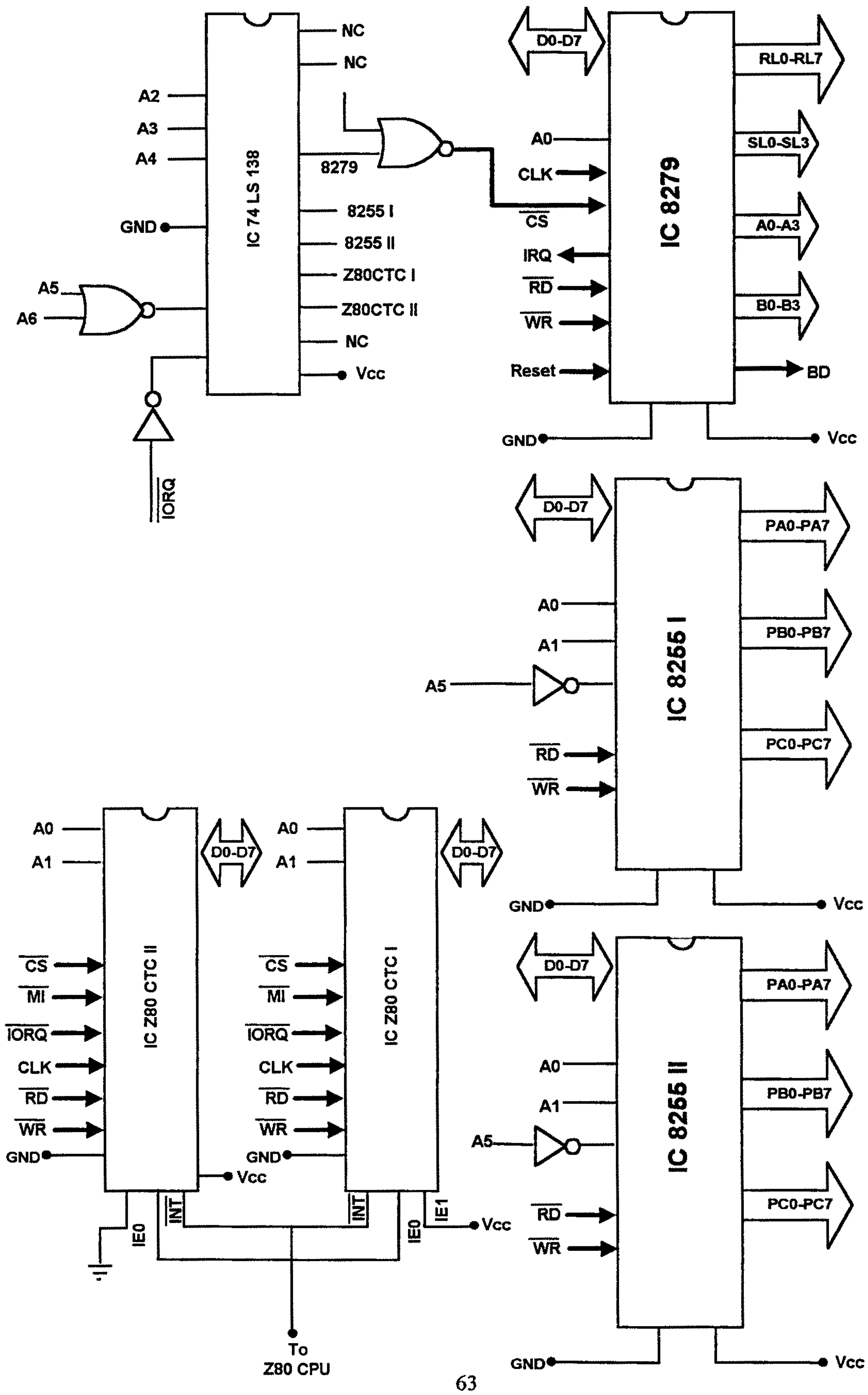


Figure 10 System Card

Here IC 74LS244 (tristate octal buffer) is used as the input port. The 8 pins to be read are connected to the input side of the IC 74LS244 and the system data lines are connected at the output. The chip enable is signal is achieved by "OR"ing the slot lone of input, chip select line, one of the card select line and the RD line. During the Read operation, the input status is directly transferred to the system data bus.

The output of the controlling element is connected to the input port through optical isolators which provides isolation as well as the conversion of 24V dc output of controlling element the isolator sends a signal to the CPU via 74LS244. When the isolators output is on, it is sensed by a coded signal from the CPU. Each terminal number of the module is assigned a number in a consecutive order. The on-off status for each number is checked on each sweep of the input scan. (REF 35,36,37)

Output Card: Output Interface

The signal from the PLC should be electrically isolated from the output device. This card is used to transfer the process data to external controlling element. The IC 74LS373 (octal latch) is used to the process data the latch enable signal is obtained by "OR"ing the slot line slot line for output, chip selecting and select line of the "WR" line. The data transfer area to the output remains latched till the next transfer.

The output of the latch is used for driving relays housed on this cards through a driver circuit. The relays (directly switched 220V AC which is available on a PBT connector

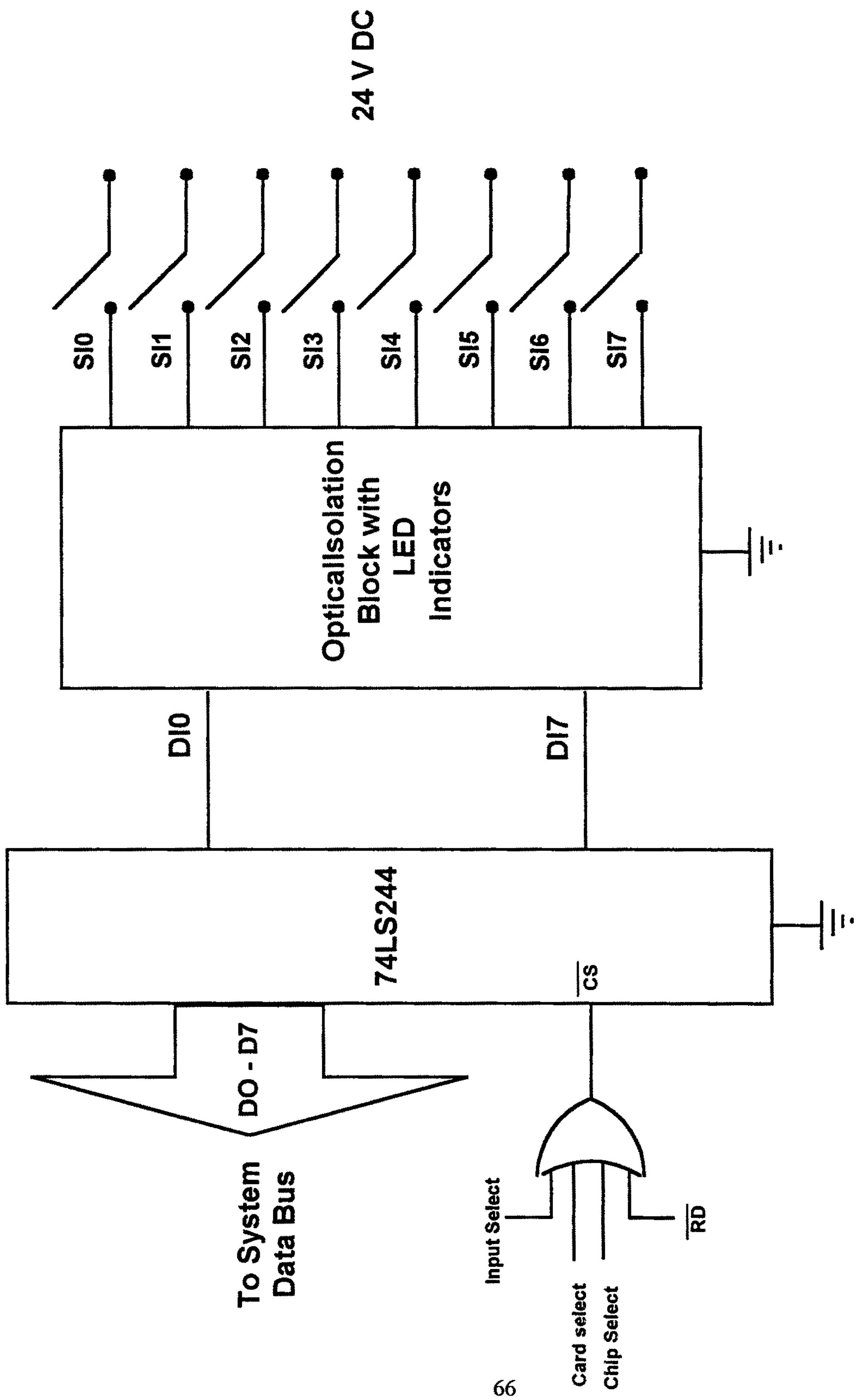


Figure 11 Input Card

connected to the output card) are used for controlling user supplied discrete (on/off) load like motor starters, solenoids valves, indicator lights etc. (REF 35,36,37)

Power Supply:

The PLC CPU must contain circuitry to convert the 220 volt AC to the required 5 volt DC values. The conversion is accomplished by built-in voltage-converting power supply. Figure 13 Power Supply Figure 13 includes the makeup of a typical power supply. The AC conditioning block is according to the Z80 CPU. A bridge rectifier is used to produce pulsing DC outputs. This dual voltage is required to operate many of the IC chips in the CPU. A regulator is used to keep the voltages at or near the 5 volt level regardless of load (CPU). [Ref 3, pp 39-40]

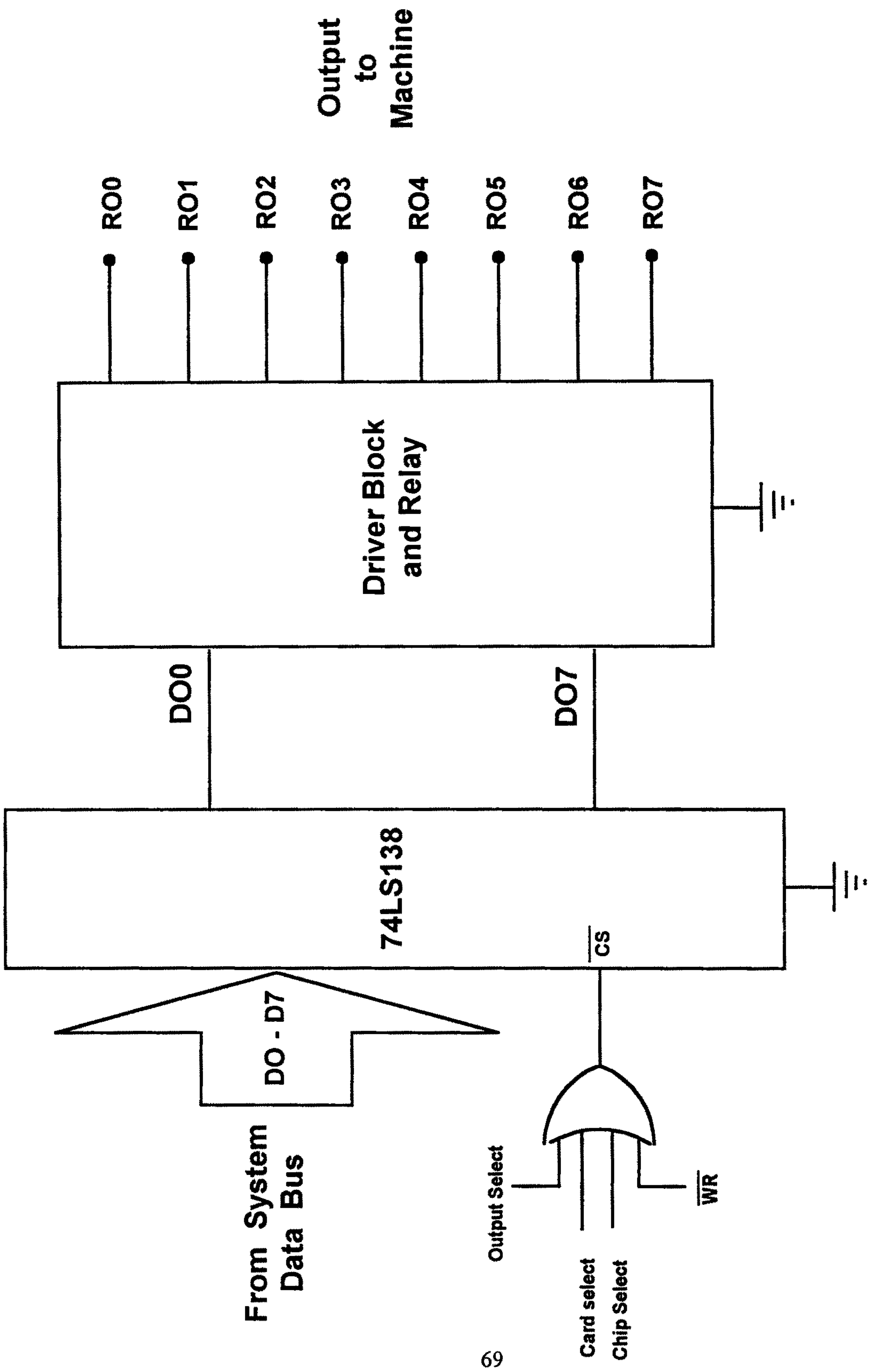


Figure 12 Output Card

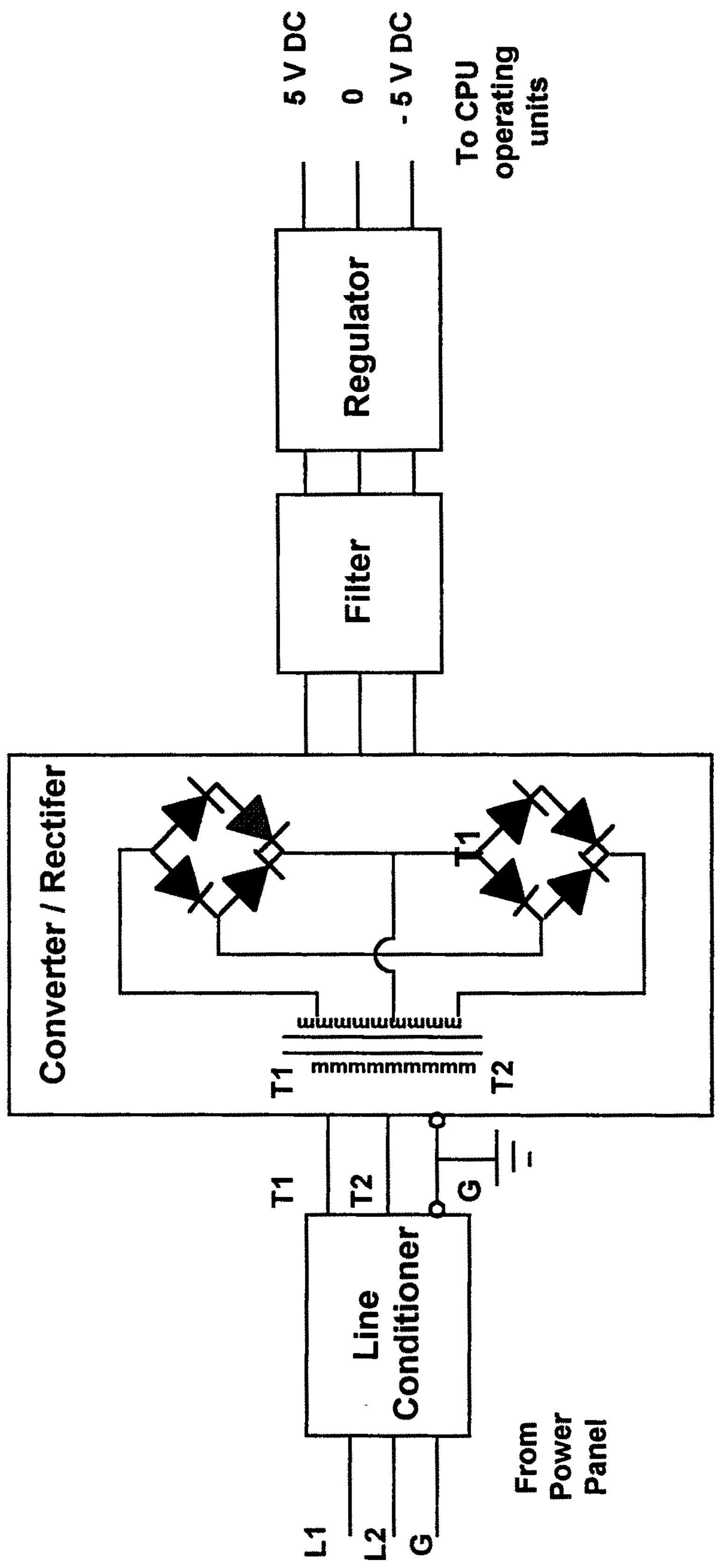


Figure 13 Power Supply

The System Software

Mathematicians are notorious for inventing new forms of mathematics for no immediate practical applications but which later prove very useful. George Boole (1815 - 1864) [Ref. 33 - chapter. 1], the mathematician and logician was no exception. He developed an algebra of sets that bears his name, Boolean algebra. The symbolism of set theory is confusing to most non mathematicians , so a complete definition of Boolean algebra and its postulates is not attempted here. It is sufficient to note that Boolean algebra provides a convenient shorthand for describing logic operations, particularly AND, OR, EXOR, NAND, NOR etc.

Using logic design using Boolean algebra, information may be represented by two-level or binary, and a combination of AND, OR and INVERT logic blocks may be used with these binary signals to perform the various functions in a computer. [Ref. 23]

This logic gate design is the heart of the programmable controller and is called a ***Binary device.*** As explained in chapter 4 there are different formats of designing the low-level language. The language used in this project is the Boolean language, which is based on the principles of Boolean algebra as discussed above.

Getting Started

Because the PLC was originally intended only to replace relay type functions, the PLCs software architecture was designed with an embedded model of the panel. A relay ladder logic (RLL) diagram consists of an arrangement of contacts and coil as in the electricians ladder diagram. This is a schematic format used to implement a combinatorial (Boolean) logic structure.

In combinatorial logic for control, the status of an output is determined by the status of a certain combination of inputs. The combinatorial logic offers little facility for describing the status of an output in relation to time, to the operational flow it the process under control or to the other output.

Special function boxes are used in RLL for control requirement that cannot be implemented with an arrangement of contacts, clever contact arrangements are used to create the interlocking logic required for event that occur sequentially in the process. A combination logic function with an RLL representation is an excellent choice is for designing the system software of the PLC system.

Scanner Generator

A PLCs is made to function by scanning its operational program. Each PLC operational cycle is made up the three separate parts input scan, program scan and output scan (as seen in chapter 4). Here we see how it can be applied to create a new algorithm.

Basic Registers

Within the PLC CPU, registers are found in two locations. The microprocessor has its internal registers, which are not directly accessible by users. But registers are needed in system programming, therefore the PLC CPU's RAM also contains slots that can be designated to hold variable information. These location or addresses becomes external registers. This project designs and uses registers of three types: *Input, Output, Accumulator register.*

Input Registers

Signal data from a specific input device is first "deposited" in the form of Os and 1s, in the input register. The input register is readily accessible to the input modules terminals or ports. The number of input registers in a PLCs normally depends on the number of input cards. The program performs this by storing from the input card in form of 1 byte data and than converting it to store in form of bits, either 1s or Os. The input data is scanned from until all the input data is scanned until all the input cards are completed.

Output registers

The output register has the same basic characteristics as the storage register. The output register is readily accessible to the output module's terminals and ports. The number of input register is equal to number of output to ports.

Accumulator Registers

The function of accumulator is similar to microprocessor internal accumulator register. It is used to perform all the PLCs functions operation such AND,LF, JMP etc.

Storage Registers

Like the input register, the storage registers are used to hold data. Conceptually, it is in the middle of the CPU (input and output register). These registers are useful to store the intermediate results of the accumulator.

Signaling output

The data associated with the output status is transferred to output terminals or ports. This is always done at end of the ladder. The process is carried out by the END instruction.

Decoding the Instruction Code Generation

Turning to code generation, the code generation phase converts the intermediate code into a sequence of machine instructions. Code generation will be applied again in the user interface software. Nevertheless this subject must be considered because a careless code generation algorithm can easily produced by an ill considered algorithm. Good code helps in reducing scan time. The method applied in machine code generation is "Microprogramming".

A group of eight bits have been used to develop a maximum of 34 instructions. The concept used here to specify whether a particular eight bit is an instruction or the data for the instruction is that of microprogramming with a few exceptions which are necessary to avoid computations, thereby increasing the scan period.

The instruction is always followed by data. During the continuous execution, the program control transfers to a particular instruction routine. This software design approach is similar to the architecture of a microprocessor where an instruction pointer points to the next instruction to be executed and the instruction decoder decodes on a combination basis to a greater extent than of more than one ladder program in the RAM and just subroutine, and then returns back to main execution.

The changing of the address is done according to instruction format. The program counter always considers the first code as instruction, once the instruction is detected, the contents of program counter is changed to the address of code and the control is transferred to that address to carry out that particular instruction. The counter is changed to the address of data, where specific status is stored. This point explains the instruction decoding. For proper decoding to take place, the instruction should be in the format given below.

Instruction Format X X DD

Where XX ➔ Respresent the instruction to be process.
DD ➔ Data

Serial Communication Service

The BIOS communication service performs RS-232C character I/O with the INT 14h serial communication DSR. The DSR provides a hardware independent interrupt driven RS-232C serial interface with more functionality than is available from DOS serial port driver.

Theory of Operation

Description : The ROM BIOS serial communication service is based on the EIA RS-232C specification and the capabilities of the National Semiconductor 8250 UART.

Data frames : The ROM BIOS serial communication service packages each data byte into a separate frame. Each frame consist of a start bit, the data bits, an optional parity bit and one or two stop bit.

INT 14h RS-232 Compatibility : INT 14h transmits data across the RS-232C I/O path in three steps.

1. The application program places the data byte to sent in AL, puts function 01h send character in AH, and performs an INT 14h.

2. The BIOS transfers the Data in AL to the serial port specified in DX. The serial of handshaking signals, such as Data Terminal Ready (DTR) and Request to Send (RST). When the external device signals that it is ready to begin the data transfer, the controller assembles the data frames and sends them across the I/O path.

3. The external device receive each charters, removes the start parity and stop bits and assembles that data bits into charters.

INT 14h RS-232 Communication DSR

Description : The INT 14h serial communication DSR provides support for sending and receiving data and determining the status of equipment used in serial communications.

The INT 14h serial communication DSR references control information stored in the BIOS data area of the system RAM. This information is located at the following offsets of segment 40h, 00h, 10h, & 7h.

The INT 14h serial communication DSR references I/O port address 3F8h - 3FFh for serial port1 and 2F8h - 2FFh for serial port2. There are no standard I/O port addresses for serial port 3-4 the addresses referenced by DSR for there optional serial ports are specific to each manufactures model. The most common I/O port address for serial port 3 are 3220h - 3227h; the most common I/O port address for serial port 4 are 3228h - 322Fh.

Invoking the INT 14h DSR : Software INT 14h invoke the INT 14h serial communication DSR. The INT 14h vector resides at address 00 : 50h in the interrupt vector table and is initialized by the BIOS to point to system ROM address F000:E739h. DOS takes over this services and re-vectors the interrupts vector table entry.

Error Handling : The serial communication service detects two kinds of errors, which are as follows:

Parameter-related errors : Parameter-related error do not return an error indication instead, the serial communication service function checks the following parameter - related conditions when it receives inputs.

- The function number specified in AH falls within the range 0-3.
- The serial port specified in DX falls within the range 0-3.
- The serial port specified in DX exists in hardware.

If any of the conditions above are not true, the serial communication service does not perform the requested function and returns with register preserved.

Time-out errors: A time-out error occurs when either a read or a write of a specified communications line was unable to occur.

The serial communication service read and write function test the line status register. When a time-out error occurs, bit 7 "time-out error" is set.

[Ref. (2 - pp. 306 - 311), (31 - pp. 258 - 270), 8, (1 - pp. 235 - 271),

Service	Description
INT 14h - 00h	Initialize the serial Adapter
INT 14h - 01h	Send Character
INT 14h - 02h	Receive Character
INT 14h - 03h	Return serial port status

Table 9 BIOS communication service for serial communication

I/O Address	R/W Status	Description
02F8h	W	Serial 2, Transmitter holding register.
02F8h	R	Serial 2, Receiver buffer register.
02F8h	R/W	Serial 2, divisor latch, low byte.
02F9h	R/W	Serial 2, divisor latch, high byte.

Table 10 I/O Address Range

The PLC Circuit Builder Program

The PLC program to define ladder diagrams is written on the Windows 95 and Windows NT 32 bit PC operating systems using Microsoft Visual C++ 4.2 compiler. The program is implemented using object oriented analysis and design. The program makes use of the Windows graphical user interface (GUI) elements to aid the user in designing the ladder diagram.

Microsoft Visual C++ 4.2 was chosen as the development tool because it offers various programmer aids like syntax colouring, integrated class hierarchy, resources and file viewer, highly sophisticated debugging, and tightly integrated help and books on line. The Appwizard application that comes with it allows an easy way to create a new application. The ClassWizard application and the dialog editor are extremely helpful to create new classes, to define Windows message handlers, and to define member variables and functions. The Microsoft Foundation Classes (MFC) cuts the development time by providing support for common dialogs, print preview support, helper classes like lists, collections, user interface classes like dialogs, windows, buttons, listboxes, comboboxes, toolbars, tooltips, statusbars, menus and scrollbars. The C++ language helps in code reuse.

System Requirements for the PLC Program

The PLC program runs on a IBM compatible PC with Windows NT or Windows 95 operating system installed. It needs the mouse as well the keyboard. A printer attached to the PC or LAN is required for getting hard copies of the ladder diagram and the instructions generated for the ladder diagram. Memory requirements are not stringent. Nevertheless, Windows 95 or Windows NT runs well with more than 8MB memory. Also a hard disk is required to install the program and store the circuits designed. Installation of the program is merely copying the executable file to the PLC directory on the hard drive.

PLC Program Capabilities

The PLC program has a user friendly graphical user interface along with all the common features of Windows based programs. Menus, Toolbars, and Keyboard Accelerators are all implemented to provide the user with alternatives to accomplish the ladder design.

The program is an MDI (Multiple Document Interface) which allows the user to view and edit multiple ladder diagrams simultaneously. Specifically, the program allows the following manipulations:

- Any number of input/output cards can be specified from 00-FF at the start
- The ladder diagram can have any number of rungs
- Parallel rungs are supported. A parallel rung is created whenever the user wants to add a component parallel to a preexisting component. The parallel rungs can have any depth. The parallel rungs are evaluated recursively. A rung can have more than one parallel rung.
- A cursor is shown to the user at all times to identify where the next component will be placed, where the parallel rung will begin from or be completed.
- The rung can have any number of components. If the rung's width is not enough to accommodate the newly drawn component, all the rungs of the ladder are automatically extended.

- The program prevents the user from adding the first switch as an output switch. No component is allowed after the Output switch. Also, the program incorporates a Start and End rung on which no other components can be placed. The cursor cannot be placed on the Start or End rung.
- If the user has already placed an End rung and then tries to add another rung, the end rung is shifted downward and a new rung is created.
- The program allows naming of all components. The names can be changed via the user interface.
- The program allows code to be generated at any stage of the ladder development. The designer can have the ladder and code documents side by side, add components to the ladder and update the code.
- All the ladder diagrams and the code can be saved to disk and loaded from the disk.
- The program allows multiple selection of objects, deleting of an object or a rung.
- The program allows printing of the ladder diagram as well as the generated code.
- When the user is creating a parallel rung, it can be aborted at any stage.
- If user moves to an upper rung and starts drawing parallel rungs the other rungs are pushed down automatically to make space.
- When extending the ladder width, the start and end objects are centered.
- The program checks for the names of switches and other components and disallows duplicate names. It also allows renaming components by double clicking on the component.

PLC Program User Interface

The program has a toolbar shown in the **Error! Reference source not found.**. It also has a menu as shown in **Error! Reference source not found.**. When a new circuit has to be designed, the user needs to use the file menu and select 'new'. Alternatively, clicking on the toolbar's first button accomplishes the same thing. The program creates a new empty circuit document. All the toolbar buttons are also equipped with tooltips which are small bubbles containing text of what the button is used for. The program also features a status bar at the bottom which is used to display messages and a vertical and horizontal grid. The status bar, the tool bar and the grid can be turned on or off using the View menu item. When a toolbar button is clicked on, the status bar also displays the function of the button.

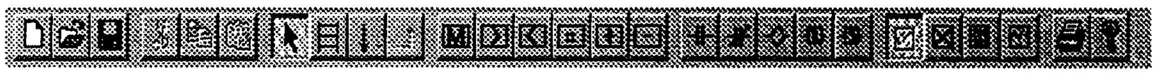


Figure 14 PLC Program Toolbar

Figure 15 PLC Program Main Menu

When a new document is opened, the user is guided to press the 'start' toolbar item or use the 'start' menu item under the Draw menu. All other toolbar buttons and menu items related to drawing ladder components are disabled at this time. When the user presses start, he is prompted for the number of input/output ports with the following dialog:

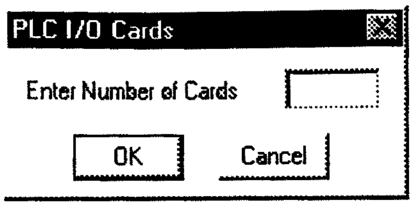


Figure 16 Dialog for Number of ports

When the user responds, the dialog validates the data and generates approproate messages if the value entered is incorrect. On successful validation, the program draws the Start component with the number of input/output ports and the first rung. The program screen looks like that in the following figure.

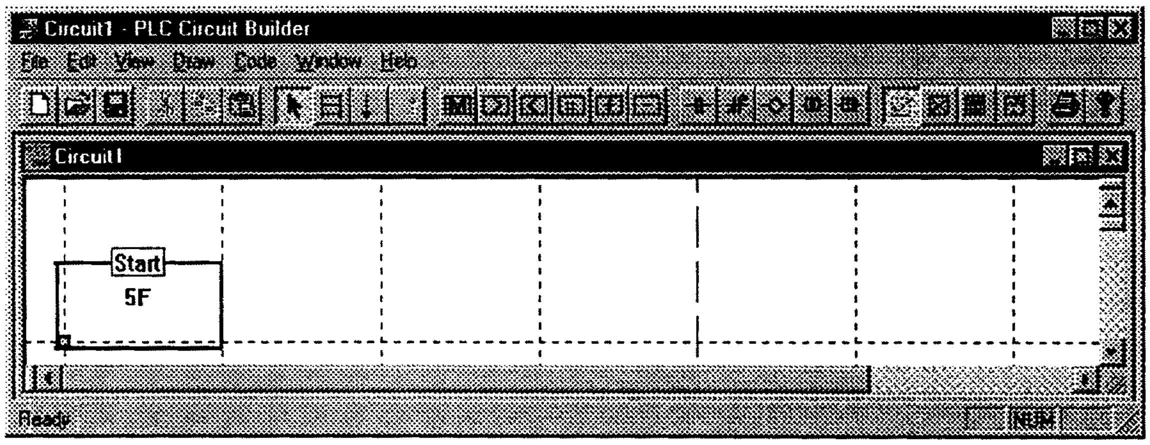

Circuit1 - PLC Circuit Builder
File Edit View Draw Code Window Help
Circuit1
Start
5F
Ready
NUM

Figure 17 PLC circuit with start rung

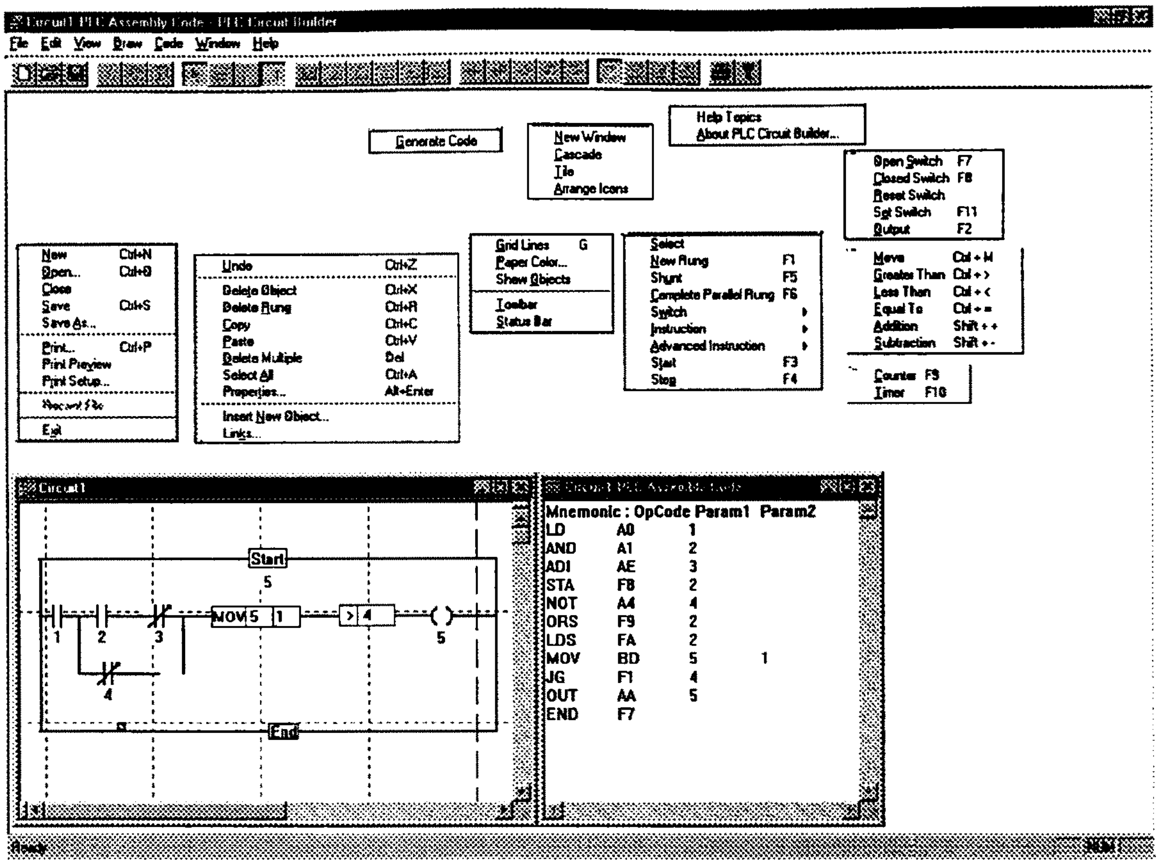
All the drawing related menu items and toolbar buttons are now enabled. The program uses a cursor to display the position of the next component which will be created. The cursor is placed at the beginning of the first rung. The user can start placing new components on the ladder with the help of the toolbar, accelerator or menu items. The following table describes the methods to create the different ladder components.

Action	Menu Item	Keyboard Accelerator	Toolbar Button
New Circuit	File/New	Ctrl+N	🗋
Open Saved Circuit	File/Open	Ctrl+O	📂
Save Circuit	File/Save	Ctrl+S	💾
Create New Rung	Draw/New Rung	F1	☰
Create a shunt or parallel rung	Draw/Shunt	F5	↓
Complete the shunt	Draw/Complete parallel rung	F6	↑
Create the Move Instruction	Draw/Instruction/Move	Ctrl+M	M
Create the Greater than Instruction	Draw/Instruction/Greater than	Ctrl + >	>
Create the Less than Instruction	Draw/Instruction/Less than	Ctrl + <	<
Create the Equal To Instruction	Draw/Instruction/Equal To	Ctrl + =	=
Create the Addition Instruction	Draw/Instruction/Addition	Shift + +	+
Create the Subtraction Instruction	Draw/Instruction/Subtraction	Shift + -	-
Draw Open Switch	Draw/Switch/Open Switch	F7	-||-

Action	Menu Item	Keyboard Accelerator	Toolbar Button
Draw Closed Switch	Draw/Switch/ Closed Switch	F8	▨
Draw Output Switch	Draw/Switch/Output	F2	▨
Draw Reset Switch	Draw/Switch/Reset Switch	-	▨
Draw Set Switch	Draw/Switch/Set Switch	F11	▨
Draw the Start Rung	Draw/Start	F3	▨
Draw the End Rung	Draw/Stop	F4	▨
Draw Counter	Draw/Advanced Instruction/Counter	F9	▨
Draw Timer	Draw/Advanced Instruction/Timer	F10	▨
Print the circuit	File/Print	Ctrl + P	▨

Table 11 Methods to create PLC ladder components

Using the GUI the user can easily navigate between menus and documents and develop the ladder diagram in very little time. The Code/Generate code menu creates the code for the ladder being seen. The following figure shows a ladder diagram with its associated code generated. The menus are also expanded for convenience.



```
Mnemonic : OpCode Param1  Param2
LD         A0     1
AND        A1     2
ADI        AE     3
STA        F8     2
NOT        A4     4
ORS        F9     2
LDS        FA     2
MOV        BD     5       1
JG         F1     4
OUT        AA     5
END        F7
```

Figure 18 PLC program screen with ladder diagram, code and all menus

Program Design

The PLC program uses the classes designed in a manner described in Figure 19. There is one instance of CPLCApp and an instance of all the tools (eg CRectTool, CStartSwitch etc) that can be used to draw different ladder components. When a new component is to be drawn, the tools are used to determine the component's attributes, bitmap etc. Since the program is a multiple document interfaces, it creates one instance of CPLCDocument for each document or ladder diagram that is being edited. Each document object creates a CPLCView object. The CPLCDocument also contains a CLadder object. The ladder object contains the CInstructRung objects. One CInstructRung object can contain another array of parallel rung related CInstructRung objects. Each CInstructRung object has an array of CInstruction object. A CInstruction object corresponds to the actual ladder diagram component shown on the screen. It also has information about the opcode of the instruction etc. The bitmaps of various drawing objects are drawn with the help of CDrawBitmap objects.

Program Code Files

The PLC program contains the following source files:

Mainfrm.Cpp, Plcdoc.Cpp, Splitfrm.Cpp, Stdafx.Cpp, Cntritem.Cpp, Rectdlg.Cpp, Plctool.Cpp, Plcobj.Cpp, Plcapp.Cpp, Plcview.Cpp, Instruct.Cpp, Instrvw.Cpp.

Apart from the source files, each source has its include file, and there are several bitmap files which define bitmaps for the PLC components. The program uses the Microsoft Foundation Class (MFC) hierarchy for the User Interface development. A brief description of the function of each file is presented.

- STDAFX.CPP: This file includes just the standard includes. A precompiled header named stdafx.pch will be generated in stdafx.obj which will contain the pre-compiled type information. This is MFC's mechanism to save compile time.
- PLCAPP.CPP: This file implements the CPLCApp object which is derived from the CWinApp MFC object. In its InitInstance method, it generates the document templates for the Instructions view document and the PLC design document. It also implements the about dialog.
- MAINFRM.CPP: This file contains the CMainFrame frame window implementation for the program.

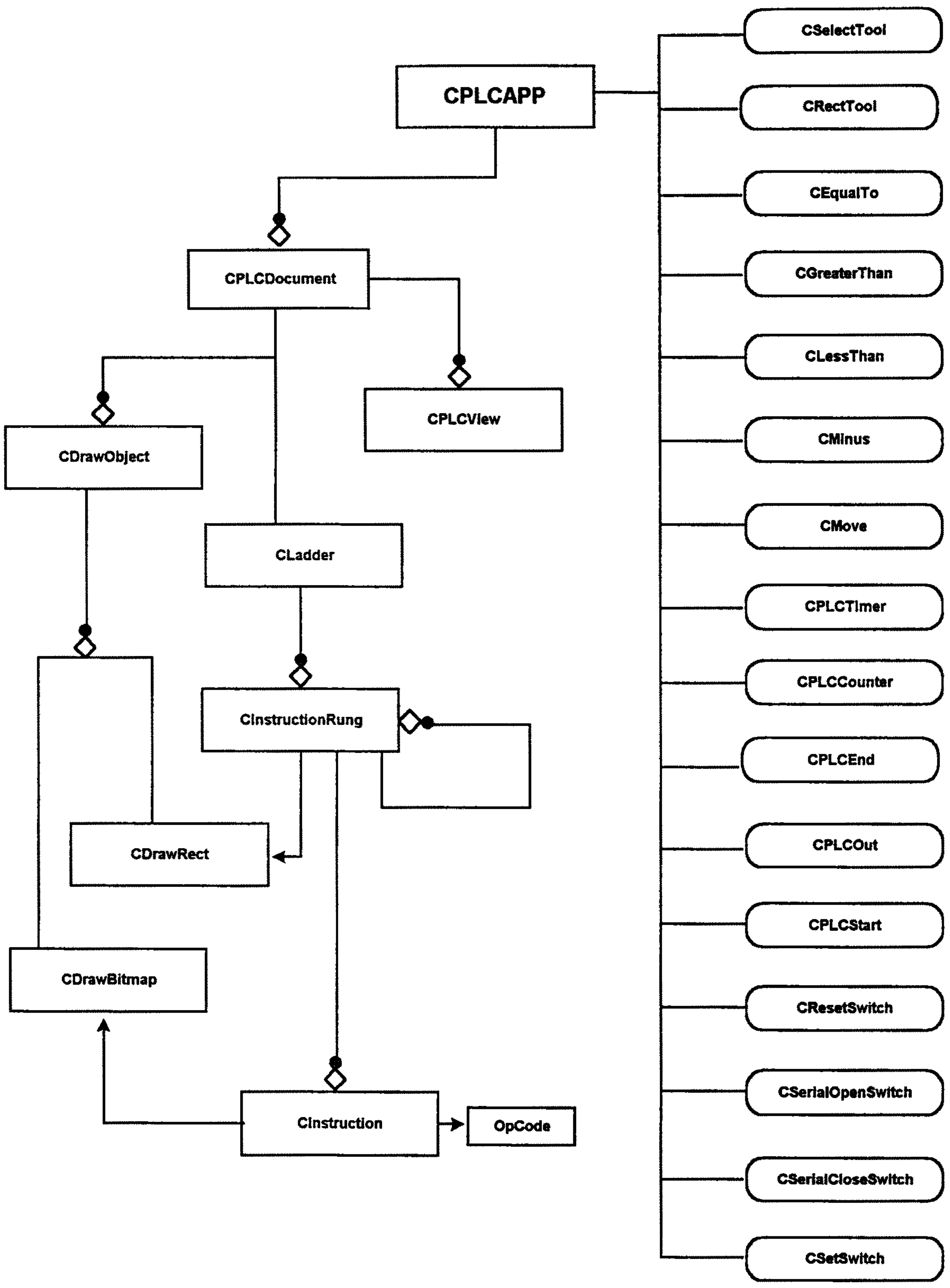


Figure 19 PLC Program Objects Interaction

- PLCDOC.CPP: This file has the drawing document class for the program
- SPLITFRM.CPP: Implements the split frame to show multiple views of the same ladder diagram.
- CNTRITEM.CPP: This has the ole container for PLC program.
- RECTDLG.CPP: This file has the dialog class for component attributes.
- PLCTOOL.CPP: All the properties of the drawing components for the ladder diagram are implemented in this file.
- PLCOBJ.CPP: This file implements all the drawing components for the ladder diagram are implemented in this file.
- PLCVIEW.CPP: The view of the ladder diagram is handled in this file. All the menu items are also handled here.
- INSTRUCT.CPP : This file handles the instruction generation objects
- INSTRVW.CPP: This file has the code to show the instructions generated for the ladder diagram.

The entire program is over 12000 lines of C++ code. It is built using a MakeFile with the Visual C++ application development environment. The program listing is attached for reference. To develop the PLC program, reference material in [Ref 36,37,38] was used.

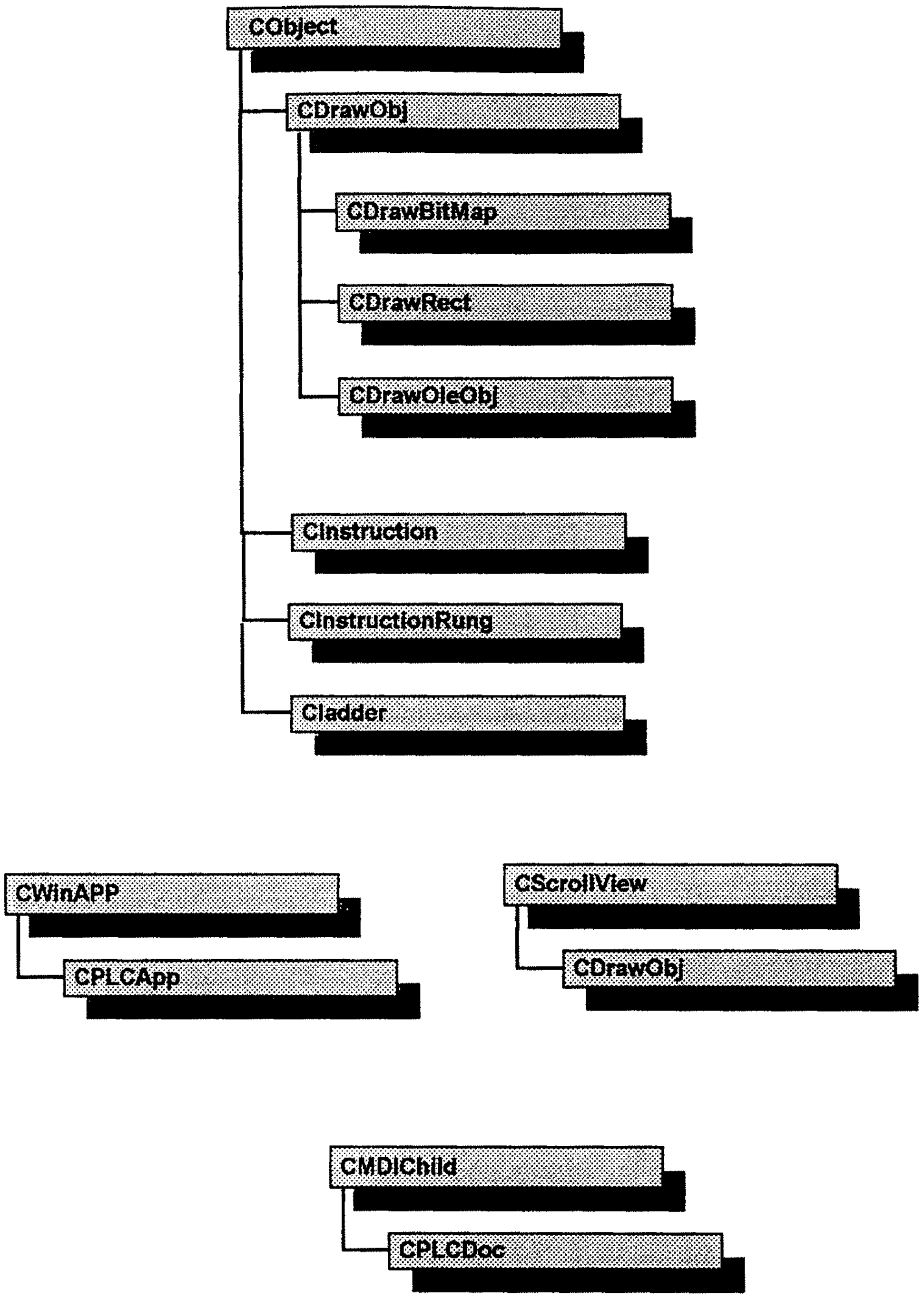


Figure 20 Class Hiierarchy Chart for PLC Program

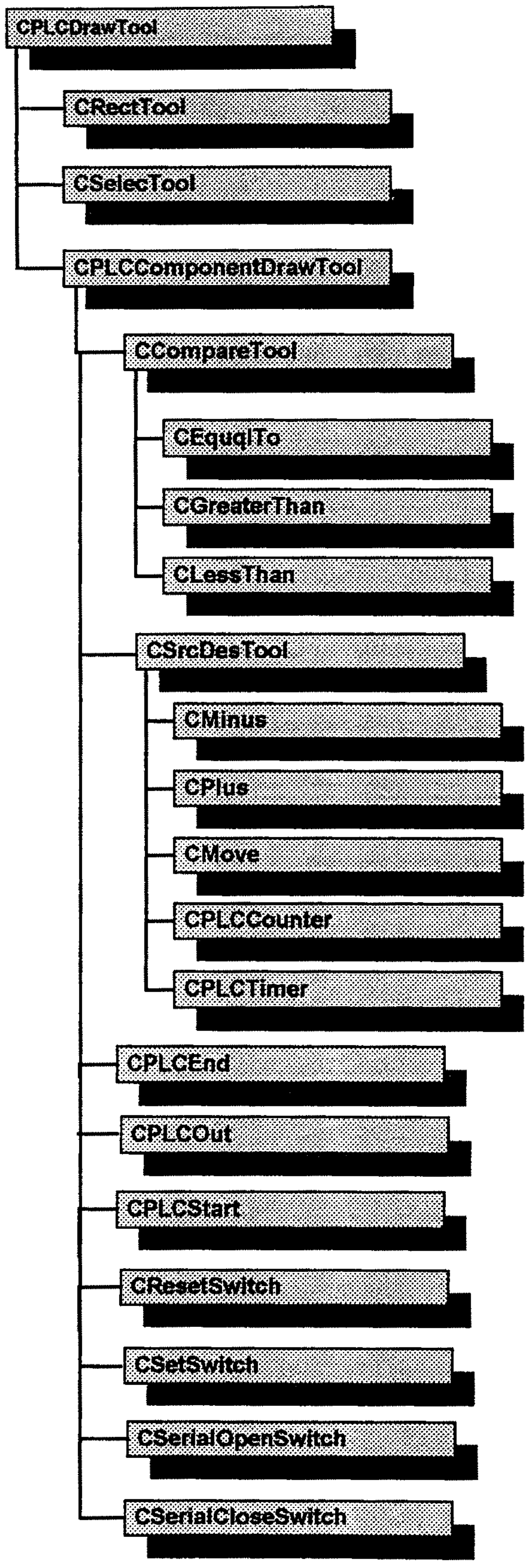

CPLCDrawTool
CRectTool
CSelecTool
CPLCComponentDrawTool
CCompareTool
CEquqlTo
CGreaterThan
CLessThan
CSrcDesTool
CMinus
CPlus
CMove
CPLCCounter
CPLCTimer
CPLCEnd
CPLCOut
CPLCStart
CResetSwitch
CSetSwitch
CSerialOpenSwitch
CSerialCloseSwitch

Figure 21 PLC Program Class Hierarchy

Appendix

Program Listing

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	OE,20	MVI C,20H	
	7E	MOVE A,M	
	87	ADD A	
	47	MOV B,A	
	E5	PUSH H	
	21,50, 22	LXI H	
	ED, 48	IN [C]	
	77	MOV M,A	
	E1	PUSH H	
	21,00,20	LXI H	
	57	MOV D,A	
	E6,01	ANDI 01	
	77	MOV M,A	
	23	INX H	
	7A	MOV A,D	
	E6,02	ANI 02	
	0F	RRC	
	77	MOV M,A	
	23	INX H	
	7A	MOV A,D	
	E6,04	ANI 04	
	0F	RRC	
	OF	RRC	
	77	MOV A,D	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	23	INX H	
	7A	MOV A,D	
	E6,08	ANI 08	
	0F	RRC	
	0F	RRC	
	0F	RRC	
	77	MOV A,D	
	23	INX H	
	7A	MOV A,D	
	E6,10	ANI 10	
	0F	RRC	
	0F	RRC	
	0F	RRC	
	0F	RRC	
	77	MOV A,D	
	23	INX H	
	7A	MOV A,D	
	E6,10	ANI 10	
	0F	RRC	
	0F	RRC	
	0F	RRC	
	0F	RRC	
	0F	RRC	
	77	MOV A,D	
	23	INX H	
	7A	MOV A,D	
	E6,20	ANI 20	
	07	RLC	
	07	RLC	
	77	MOV A,D	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	23	INX H	
	7A	MOV A,D	
	E6,20	ANI 20	
	07	RLC	
	07	RLC	
	77	MOV A,D	
	23	INX H	
	7A	MOV A,D	
	E6,20	ANI 20	
	07	RLC	
	77	MOV M,A	
	E1	POP H	
	23	INX H	
	0C	INR C	
	05	DCR B	
	C2,0B,20	JNZ X1	
	E1	POP H	
	23	INX H	
	7E	MOV A,M	
	C3	JMP BBB	
	23	INX H	
	C3	JMP AAA	
	FE,A0	CPI AO	Detection of Instruction
	CA	JZ	
	FE,A0	CPI A1	
	CA	JZ	
	FE,A0	CPI A2	
	CA	JZ	
	FE,A0	CPI A3	
	CA	JZ	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	FE,A0	CPI A4	
	CA	JZ	
	FE,A0	CPI A5	
	CA	JZ	
	FE,A0	CPI A6	
	CA	JZ	
	FE,A0	CPI A7	
	CA	JZ	
	FE,A0	CPI A8	
	CA	JZ	
	FE,A0	CPI A9	
	CA	JZ	
	FE,A0	CPI AA	
	CA	JZ	
	FE,A0	CPI AB	
	CA	JZ	
	FE,A0	CPI AC	
	CA	JZ	
	FE,A0	CPI AD	
	CA	JZ	
	FE,A0	CPI AE	
	CA	JZ	
	FE,A0	CPI AF	
	CA	JZ	
	FE,A0	CPI B0	
	CA	JZ	
	FE,A0	CPI B1	
	CA	JZ	
	FE,A0	CPI B2	
	CA	JZ	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	FE,A0	CPI B3	
	CA	JZ	
	FE,A0	CPI B4	
	CA	JZ	
	FE,A0	CPI B5	
	CA	JZ	
	FE,A0	CPI B6	
	CA	JZ	
	FE,A0	CPI B7	
	CA	JZ	
	FE,A0	CPI B8	
	CA	JZ	
	FE,A0	CPI B9	
	CA	JZ	
	FE,A0	CPI BA	
	CA	JZ	
	FE,A0	CPI BB	
	CA	JZ	
	FE,A0	CPI BC	
	CA	JZ	
	FE,A0	CPI BD	
	CA	JZ	
	FE,A0	CPI BE	
	CA	JZ	
	FE,A0	CPI BF	
	CA	JZ	
	FE,A0	CPI C0	
	CA	JZ	
	FE,A0	CPI C1	
	CA	JZ	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
A0	23	INX H	LOAD INSTRUCTION
	56	MOV D,M	
	E5	PUSH H	
	21, A.M	LXI H, A.M	
	E5	PUSH H	
	6A	MOV L,D	
	26,10	MVI H,20	
	46	MOV B,M	
	E1	POP H	
	70	MOV B,M	
	E1	POP H	
	C3, NLA	JMP NLA	
A1	23	INX H	AND INSTRUCTION
	56	MOV D,M	
	E5	PUSH H	
	21, A.M	LXI H, A.M	
	E5	PUSH H	
	6A	MOV L,D	
	26,10	MVI H,20	
	46	MOV B,M	
	E1	POP H	
	7E	MOV A,M	
	A0	ANA B	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP NLA	
A2	23	INX H	OR INSTRUCTION
	56	MOV D,M	
	E5	PUSH H	
	21, A.M	LXI H, A.M	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	E5	PUSH H	
	6A	MOV L,D	
	26,10	MVI H,20	
	46	MOV B,M	
	E1	POP H	
	7E	MOV A,M	
	B0	ORA B	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP: NLA	
A3	23	INX H	EXOR INSTRUCTION
	56	MOV D,M	
	E5	PUSH H	
	21, A.M	LXI H, A.M	
	E5	PUSH H	
	6A	MOV L,D	
	26,10	MVI H,20	
	46	MOV B,M	
	E1	POP H	
	7E	MOV A,M	
	A8	XRA B	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP: NLA	
A4	E5	PUSH H	NOT INSTRUCTION
	21, AM	LXI, AM	
	37	STC	
	3F	CMC	
	7E	MOV A,M	
	1F	RAR	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	3F	CMC	
	17	RAL	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP: NLA	
A5	23	INX H	STORE INSTRUCTION
	56	MOV D,M	
	E5	PUSH H	
	21, A.M	LXI H, A.M	
	7E	MOV A,M	
	6A	MOV L,D	
	26,21	MVI H,21	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP: NLA	
A6	23	INX H	LOAD FROM STORE MEMORY
	56	MOV D,M	
	E5	PUSH H	
	6A	MOV L,D	
	26,21	MVI H,21	
	7E	MOV A,M	
	21, A.M	LXI H, A.M	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP: NLA	
A7	23	INX H	AND WITH STORE MEMORY
	56	MOV D,M	
	E5	PUSH H	
	6A	MOV L,D	
	26, S.M	MVI H, S.M	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	7E	MOV A,M	
	21, A.M	LXI H, A.M	
	A6	ANA M	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP: NLA	
A8	23	INX H	OR WITH STORE MEMORY
	56	MOV D,M	
	E5	PUSH H	
	6A	MOV L,D	
	26, S.M	MVI H, S.M	
	7E	MOV A,M	
	21, A.M	LXI H, A.M	
	B6	ORA M	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP: NLA	
AA	23	INX H	OUTPUT TO OUTFILE
	56	MOV D,M	
	E5	PUSH H	
	21, A.M	LXI H, A.M	
	7E	MOV A,M	
	FF,00	CPI 00H	
	CA	JZ : Z1	
	3E, FF	MVI A,FF	
Z1	6A	MOV L,D	
	26,	MVI H ,O.M	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP : NLA	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
AB	E5	PUSH H	OUT TO OUPUT PORTS
	21,O.M	LXI,O.M	
	E5	PUSH H	
	06,80	MVI B,80	
	16,01	MVI D,01	
Z2	7E	MOV A,M	
	A2	ANA D	
	77	MOV M,A	
	23	INX H	
	CB,02	RLC D	
	05	DCR B	
	C2, Z2	JNZ: Z2	
	E1	POP H	
	21,PO.M	LXI D,PO.M	
	OE,08	MVI C,08	
Z3	06,08	MVI,08	
	7A	MOV A,M	
Z4	23	INX H	
	B6	ORA M	
	05	DCR B	
	C2,Z4	JNZ: Z4	
	EB	XCH G	
	77	MOV M,A	
	EB	XCHG	
	23	INX H	
	13	INX D	
	0D	DCR C	
	C2,Z3	JNZ: Z3	
	21,PO.M	LXIH,PO.M	
	OE,40	MVI C,40H	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	06,08	MVI B,80	
Z5	7E	MOV A,M	
	ED,49	OUT [C]	
	OC	INR C	
	23	INX H	
	05	DCR B	
	C2,Z5	JNZ: Z5	
	E1	POP H	
	C3,Z0	JMP: Z0	
AC	23	INX H	NOT IMMEDIATE LOAD STATUS
	56	MOV D,M	
	E5	PUSH H	
	21, A.M	LXI H, A.M	
	E5	PUSH H	
	6A	MOV L,D	
	26, A.M	MVI H, A.M	
	7E	MOV A,M	
	37	STC	
	3F	CMC	
	1F	RAR	
	3F	CMC	
	17	RAL	
	E1	POP H	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP: NLA	
AD	23	INX H	NOR LOAD STATUS INSTRUCTION
	56	MOV D,M	
	E5	PUSH H	
	21, A.M	LXI H, A.M	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	E5	PUSH H	
	6A	MOV L,D	
	26, A.M	MVI H, A.M	
	46	MOV B,M	
	E1	POP H	
	7E	MOV A,M	
	B0	ORA B	
	37	STC	
	3F	CMC	
	1F	RAR	
	3F	CMC	
	17	RAL	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP: NLA	
AE	23	INX H	NAND LOAD STATUS INSTRUCTION
	56	MOV D,M	
	E5	PUSH H	
	21, A.M	LXI H, A.M	
	E5	PUSH H	
	6A	MOV L,D	
	26, A.M	MVI H, A.M	
	46	MOV B,M	
	E1	POP H	
	7E	MOV A,M	
	A0	ANA B	
	37	STC	
	3F	CMC	
	1F	RAR	
	3F	CMC	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	17	RAL	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP: NLA	
AF	23	INX H	NOT IMMEDIATE STORE MEMORY
	56	MOV D,M	
	E5	PUSH H	
	6A	MOV L,D	
	26, S.M	MVI H,S.M	
	7E	MOV A,M	
	37	STC	
	3F	CMC	
	1F	RAR	
	3F	CMC	
	17	RAL	
	21, A.M	LXI, A.M	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP: NLA	
B0	23	INX H	NAND STORE MEMORY INSTRUCTION
	56	MOV D,M	
	E5	PUSH H	
	6A	MOV L,D	
	26, A.M	MVI H, A.M	
	7E	MOV A,M	
	21,A.M	LXI H,A.M	
	A6	ANA M	
	37	STC	
	3F	CMC	
	1F	RAR	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	3F	CMC	
	17	RAL	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP: NLA	
B1	23	INX H	NOR STORE MEMORY INSTRUCTION
	56	MOV D,M	
	E5	PUSH H	
	6A	MOV L,D	
	26, A.M	MVI H, A.M	
	7E	MOV A,M	
	21,A.M	LXI H,A.M	
	B6	ORA M	
	37	STC	
	3F	CMC	
	1F	RAR	
	3F	CMC	
	17	RAL	
	77	MOV M,A	
	E1	POP H	
	C3, NLA	JMP: NLA	
B2	23	INX H	COMPARE IMMEDIATE INSTRUCTION
	46	MOV B,M	
	E5	PUSH H	
	21,A.M	LXI H,AM	
	4E	MOV C,M	
	E1	POP H	
	C3, NLA	JMP: NLA	
B3	23	INX H	COMPARE WITH INPUT STATUS
	56	MOV D,M	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	E5	PUSH H	
	6A	MOV L,D	
	26, I.M	MVI H, I.M	
	46	MOV B,M	
	21,A.M	LXI A.M	
	4E	MOV C,M	
	E1	POP H	
	C3, NLA	JMP: NLA	
B4	23	INX H	COMPARE WITH STORE MEMORY
	56	MOV D,M	
	E5	PUSH H	
	6A	MOV L,D	
	26, S.M	MVI H, S.M	
	46	MOV B,M	
	21,A.M	LXI A.M	
	4E	MOV C,M	
	E1	POP H	
	C3, NLA	JMP: NLA	
B5	23	INX H	JUMP IF EQUAL
	5E	MOV E,M	
	23	INX H	
	56	MOV D,M	
	79	MOV A,C	
	B8	CMP B	
	C2,NLA	JNZ: NLA	
	EB	XCHG	
	C3, AAA	JMP: AAA	
B6	23	INX H	JUMP IF LESS
	5E	MOV E,M	
	23	INX H	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	56	MOV D,M	
	79	MOV A,C	
	B8	CMP B	
	D2, NLA	JNC: NLA	
	EB	XCHG	
	C3, AAA	JMP: AAA	
B7	23	INX H	JUMP IF GREATER
	5E	MOV E,M	
	23	INX H	
	56	MOV D,M	
	79	MOV A,C	
	B8	CMP B	
	F5	PUSH PSW	
	C1	POP B	
	79	MOV A,C	
	E6,41	ANI 41H	
	FE,00	CPI 00	
	C2, NLA	JNC: NLA	
	EB	XCHG	
	C3, AAA	JMP: AAA	
B8	23	INX H	SET IMMEDIATE INSTRUCTION
	56	MOV D,M	
	E5	PUSH	
	21,I.M	LXI I.M	
	7E	MOV A,M	
	FE 01	CPI 01H	
	C2, Z10	JNZ: Z10	
	23	INX H	
	E5	PUSH H	
	6E	MOV L,M	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	26,O.M	MVI H,O.M	
	3E,01	MVI A,01	
	77	MOV M,A	
	E1	POP H	
	E1	POP H	
	23	INX H	
Z10	C3, NLA	JMP: NLA	
B9	23	INX H	RESET IMMEDIATE INSTRUCTION
	56	MOV D,M	
	E5	PUSH	
	21,I.M	LXI I.M	
	7E	MOV A,M	
	FE 01	CPI 01H	
	C2, Z11	JNZ: Z11	
	23	INX H	
	E5	PUSH H	
	6E	MOV L,M	
	26,O.M	MVI H,O.M	
	3E,01	MVI A,01	
	77	MOV M,A	
	E1	POP H	
	E1	POP H	
	23	INX H	
Z10	C3, NLA	JMP: NLA	
BA	E5	PUSH	RESET THE COUNTER
	21,A.M	LXI A.M	
	7E	MOV A,M	
	23	INX H	
	77	MOV M,A	
	E1	POP H	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	C3, NLA	JMP: NLA	
BB	23	INX H	LOAD THE COUNTER
	56	MOV D,M	
	E5	PUSH H	
	6A	MOV L,D	
	26,C.M	MVI H,C.M	
	46	MOV B,M	
	21, A.M	LXI H, A.M	
	70	MOV M,B	
	E1	POP H	
	C3, NLA	JMP: NLA	
BB	23	INX H	COUNTER INSTRUCTION
	56	MOV D,M	
	23	INX H	
	5E	MOV E,M	
	E5	PUSH H	
	6A	MOV L,D	
	26,C.M	MVI H,C.M	
	46	MOV B,M	
	E5	PUSH H	
	7A	MOV A,D	
	C6 ,64	ADI 64	
	6F	MOV L,A	
	7E	MOV A,M	
	70	MOV M,B	
	FE 01	CPI O1	
	CA,J1	JZ: J1	
	21,RST	LXI H,RST	
	7E	MOV A,M	
	FE,01	CPI 01	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	C2,J2	JNZ: J2	
	E1	POP H	
	73	MOV M,E	
	E5	PUSH H	
	6A	MOV L,D	
	26,CFP	MVI H,CFP	
	3E,00	MVI A,00	
	77	MOV M,A	
J2	21,A.M	LXI H,A.M	
	7E	MOV A,M	
	FE,01	CPI 01	
	C2,J3	JNZ: J3	
	E1	POP H	
	35	DCR M	
	7E	MOV A,M	
	FE,00	CPI 00	
	C2,J3	JNZ: J3	
	6A	MOV L,D	
	26,CFP	MVI H,CFP	
	3E,01	MVI A,01	
	77	MOV M,A	
J3	E1	POP H	
	C3,NLA	JMP: NLA	
J1	6A	MOV L,D	
	26,CFP	MVI H,CFP	
	3E,01	MVI A,01	
	77	MOV M,A	
	E1	POP H	
	73	MOV M,E	
	C3,J3	JMP: J3	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
BC	23	INX H	MOVE INSTRUCTION
	56	MOV D,M	
	23	INX H	
	4E	MOV C,M	
	6A	MOV L,D	
	26,I.M	MIV H,I.M	
	46	MOV B,M	
	6B	MOV L,C	
	7E	MOV A,M	
	23	INX H	
	56	MOV D,M	
	23	INX H	
	46	MOV C,M	
	6B	MOV L,C	
	26,I.M	MIV H,I.M	
	7E	MOV A,M	
	6A	MOV L,D	
	77	MOV M,A	
	C3,NLA	JMP: NLA	
BD	23	INX H	ADD INSTRUCTION
	56	MOV D,M	
	23	INX H	
	4E	MOV C,M	
		PUSH H	
		LXI H, A.M	
		MOV A,M	
		CPI 01	
		JNZ : J11	
	6A	MOV L,D	
	26,I.M	MIV H,I.M	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	46	MOV B,M	
	6B	MOV L,C	
	***	ADD B	
	FE,11H	CPI 11H	
	CA,J11	JZ : J12	
	35	DCR M	
	77	MOV M,A	
	23	INX H	
J11	C3,NLA	JMP: NLA	
J12	35	DCR H	
	****	MOV M,01	
	23	INX H	
	C3,NLA	JMP: NLA	
BD	23	INX H	SUB INSTRUCTION
	56	MOV D,M	
	23	INX H	
	4E	MOV C,M	
		PUSH H	
		LXI H, A.M	
		MOV A,M	
		CPI 01H	
		JNZ : J13	
		MOV L,D	
		MVI H, I.M	
		MOV L,C	
		MOV A,M	
	***	SUB B	
	E5	POP H	
	***	DCR H	
	77	MOV M,A	

LABEL	OP.CODE	MNEMONICS	COMMENTS
	31,FF,27	LXI SP	Inputting the input status
	21,00,30	LXI H	
	23	INX H	
J13	C3,NLA	JMP: NLA	

Data Operator Instructions

These instruction provide the PLC with the capability with various data operation. These operation are done with 256 memory location of input port.This is achieved in our PLC using instruction function.

Format **Fn _ _**

Where first digits represent the function number which dinotes functional data operation id to be done.

Fn A0 - LD

Fn A0 _ _

This is to 2 byte instruction that load the accumulator with the contents of inputs memory specified by the following byte. The input memory varies from (00 -FF).

Fn A1 - AND

Fn A1 _ _

This is to 2 byte instruction that perform bit by bit logical AND operation on the contents of the accumlator, using the contents of the input memory specified by the 1 byte.

Fn A2 - OR

Fn A2 _ _

All features are same as Fn A0 except that it performs logically OR operation.

Fn A3 - EXOR

Fn A3 _ _

All features are same as Fn A0 except that it performs logically EXOR operation.

Fn A3 - NOT

Fn A4

This is instruction that causes negation of the Boolean accumulator content.

Fn A5 - STA

Fn A5 _ _

This is two byte instruction that loads from the Boolean accumulator with the content of the store memory specified by the following one byte.

Fn A6 - LDS

Fn A6 _ _

This is two byte instruction that loads the Boolean accumulator with the content of the store memory specified by the following one byte.

Fn A7 - ANDS

Fn A7 _ _

This is two byte instruction that performs bit by bit logically AND operation on the contents of the Boolean accumulator, using the contents of the store memory specified by the following one byte.

Fn A8 - ORS

Fn A3 _ _

All the features are same as Fn A7 expect that it performs logically OR operation.

Fn A9 - EXORS

Fn A9 _ _

All the features are same as Fn A7 expect that it performs logically OR operation.

Fn AA - OUT

Fn AA _ _

This is two byte instruction that loads the content of the Boolean accumulator to the output memory.

Fn AB - END

Fn AB

This is one instruction that outs the content of the output memory to the output ports.

Fn AC - NTI

Fn AC _ _

This is one byte instruction that causes negation of the specified input memory location.

Fn AD - NOR

Fn AD _ _

This is two byte instruction that performs logically NOR operation on the content of Boolean accumulator using the content of specified input memory location.

Fn AE - NAND

Fn AE _ _

All the features are same as Fn AD expect that it performs logically NAND operation.

Fn AF - NTS

Fn AF _ _

All the features are same as Fn AC expect that it performs negation with the store memory location.

Fn B0 - NANDS

Fn BO _ _

All the features are same as Fn AE expect that it performs logically NAND with the store memory location.

Fn B1 - ORS

Fn B1 _ _

All the features are same as Fn AD expect that it performs logically NOR with the store memory location.

Fn B2 - CPI

Fn B2 _ _

This is two byte instruction that compares the contents of the accumulator with the immediate data.

Fn B3 - CMPR

Fn B3 _ _

This is two byte instruction that compares the contents of the Boolean accumulator with the of the input memory specified by following one byte.

Fn B4- CMPRS

Fn B4_ _

All feature are same as Fn B3 except that ot performs the compare instructions with the stored memory location.

Fn B5 - JE

Fn B5 _ _ _ _

This is four byte instruction. This conditional instruction if the condition for equal is satisfied the instructions loads the program counter with contents of memory location. The instruction is fetched from the new location loaded into the program counter.

Fn B6 - JL

Fn B6_ _ _ _

All feature are same as Fn B5 except that it performs for jump if less than condition is satisfied.

Fn B7 - JG

Fn B7_ _ _ _

All feature are same as Fn B5 except that it performs for jump if greater than condition is satisfied.

Fn B8 - SET

Fn B8 _ _

This is two byte instruction. It is used as an auxiliary relay. It sets the specified location.

Fn B9 - RESET

Fn B9_ _

All feature are same as Fn B9 except that it performs for reset operation.

Fn BA - RTC

Fn BA _

This is one byte instruction used to reset the counter.

Fn BB - LDC

Fn BB _ _

This is two byte instruction it performs operation of load the specified counter.

Fn BC - COUNT

Fn BC _ _ _ _

This is four byte instruction. It provides the count, specified by the following byte. The counter varies from 00 to FF.

Assembler Flowcharts

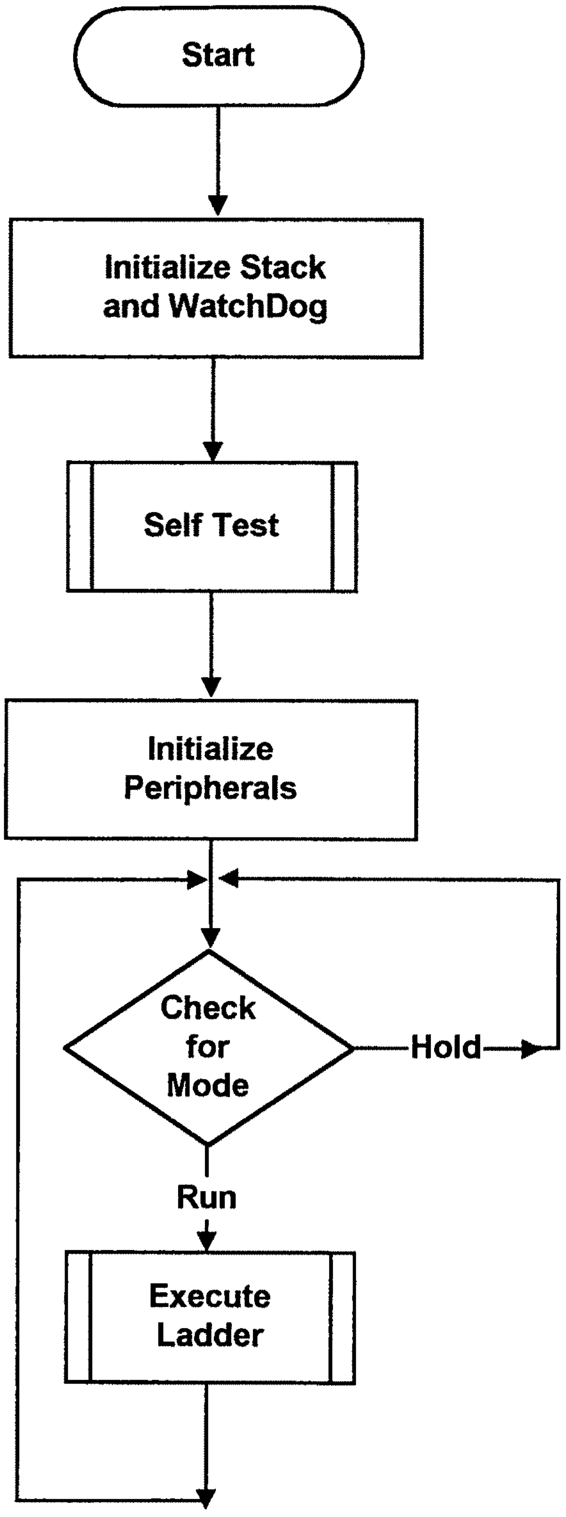


Figure 22 Flowchart 1

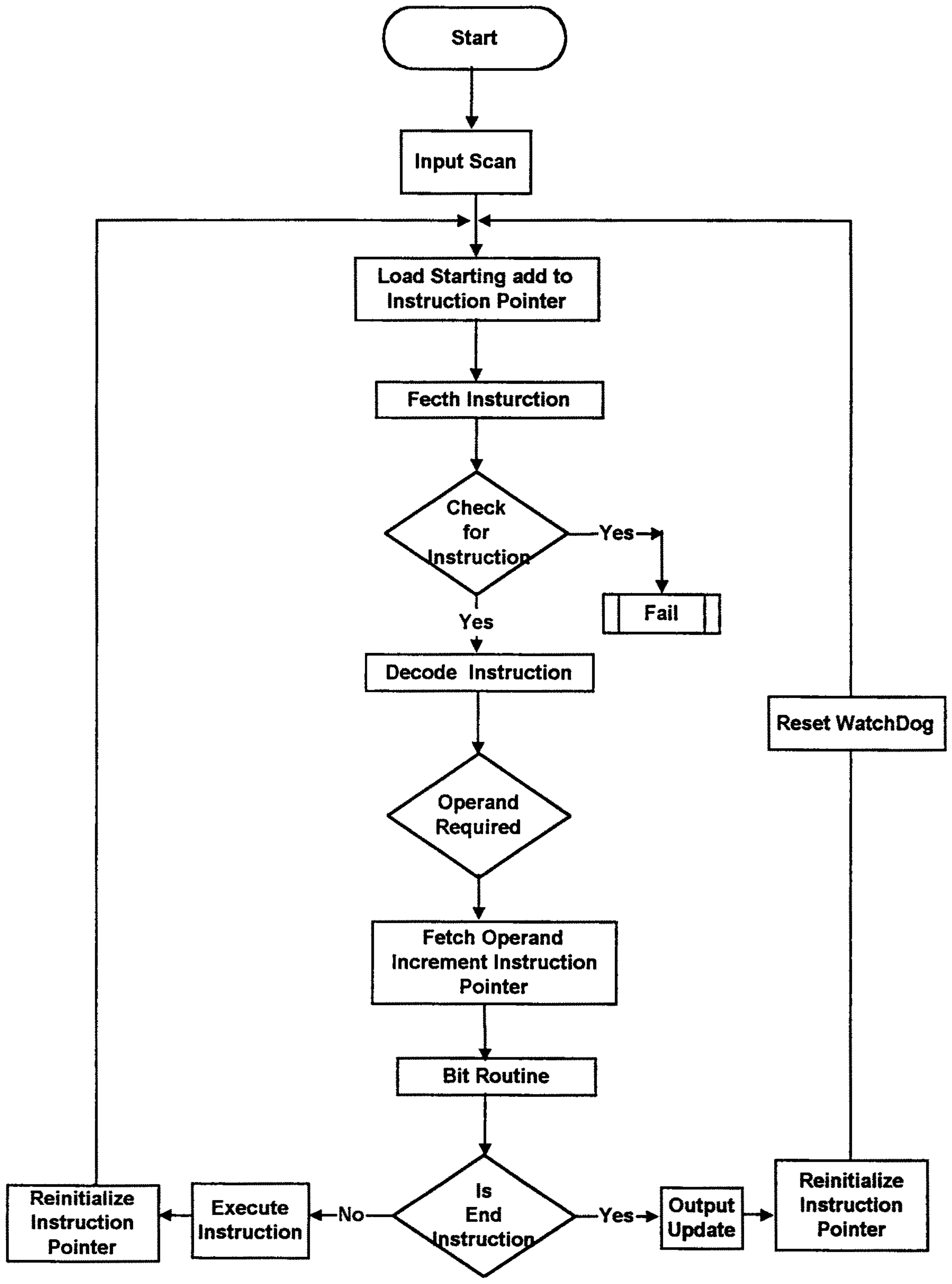


Figure 23 Flowchart 2

Typical Performance Characteristics

Introduction

The controller built around Z80 CPU can handle up to 128 inputs and 64 output with further expansion resulting in the total handling capacity of 4098 I/O ports.

Input

The input module interface on /off signals to the controller from equipment such as pushbuttons, lamps, limit switches, relay and contacts. The modules provide a high noise immunity. Interface using optical isolators and insulators for each plant signal. An indicator lamp is provided for the state of each channel.

The output module interfaces with the process data to the external control element through a driver stage and relay. The relays are used for controlling user supplied discrete (ON/OFF) load like motor starter, valves, indicator lights etc. An indicator lamp is provided for the state of each channel.

Communication

The RS232 serial port interface to the controller is used for the online programming of the controller from the PC using the PLC program exclusively for the ladder logic programming and compiling and transfer to the machine code over the RS232C serial interface.

Memory Option

The controller has 2K RAM for monitor program, 4K RAM for the loading of ladder diagram of which 2K is for the ladder program. Over and above space is available for further expansion.

Programming

The controller can be programmed from an external keyboard or can be online programmed from the PC using the PLC program developed for this purpose.

Program Capacity

The capacity depends on the memory size and the number of data tables. The 64K memory can accommodate approximately 5000 instructions.

Construction

This ruggedly constructed controller is designed for harsh industrial environments. A range of modules plug into a 19 inch rack mounting metal subtrack. The plug in modular construction allows a module substitution approach for repair maintenance and easy upgrading of the controller.

Application

The controller consists of power supply, memory, input processor, ladder program mounted in a subtrack with facilities for additional modules and equipment. These controller are well suited to applications in complex industrial plants.

Features

Automatic self testing, memory size field upgradable, online programming from PC, comprehensive instruction set.

Specifications Summary

I/O	Basic I/O	128 up to 2048 Input points 64 up to 2048 Output points
	Serial Link	1 point
Memory	User memory	4K
	Total Capacity	64K
Program	Monitor	2K
	Ladder program	2K (500 instruction)
Power Supply	220 V, 50Hz, single phase	
Input Rating	24 V DC, 20 mA (max)	
Output Rating	220 V AC, 50Hz, 1A(max)	
Environment	Temperature	0-60 degree C operating -20 degree to 70 degree storage
	Humidity	10 to 90% noncondensing
Mounting	16 inch (483mm) rack mounting	
Additional Facilities	On line program charges, Remote programming, Flexible self test, Automatic self test, Memory size upgrading	

References

[1] G. Ian Warnock, " Programmable Controllers" Library of Congress Cataloging-in-Publication Data. Prentice Hall International (UK) Ltd, Second Edition 1992.

[2] L. George and JR. Batten "Programmable Controllers: Hardware, Software & Application" Library of Congress Cataloging-in-Publication Data. McGraw-Hill, Inc Book Company. Second Edition 1994.

[3] John W. Webb and Ronald A. Reis "Programmable Logic Controllers: Principles and Application" Libraray of Congress Cataloging-in-Publication Data. Prentice Hall Book Company, Inc. Third Edition 1995.

[4] Kissell E. Thomas "Understanding and Using Programmable Controllers" Lib of Congress Cataloging-in-Publication. Prentice Hall Inc. First Edition 1986.

[5] Richard A. Cox " Technician's Guide To Programmable Controllers". Library of Congress Cataloging-in-Publication. Delmar Publishers, Inc. Second Edition 1989.

[6] Frank D. Petruzella "Programmable Logic Controllers" Library of Congress Cataloging-in-Publication. McGraw-Hill Book Company. Library of Congress Cataloging-in-Publication. First Edition 1989.

[7] I. Scott Mackengic "The 8051 Microcontroller" Library of Congress Cataloging-in-Publication. Prentice Hall, Inc. Second Edition 1995.

[8] Douglas M. Linsidine "Control System Process / Industrial Instruments and controls". Handbook fourth Edition McGraw-Hill 1993.

[9] Mahmond S. Magdi "Computer Operated System Control" Marcel Dekke, Inc. 1991.

[10] Kennedy R. H "Selecting Temperture Sensors" Chemical Engineer. Aug 1983.

[11] Burke A. "Linearizing with a single Resistor Electronics" June 2,1981.

[12] JMC Dermott "Sensors and Transducer" EDN March 20, 1980.

[13] Renn Radnsor "Instrument Engineers" Handbook rev. Edition Chilton Book Company, 1982.

[14] Cohen E.M and Febervari W. "Sequential Control" Chemical Engineering. 29/4/79.

[15] W. F. Stoecker and P. A. Stoecker "Microcomputer Control of Thermal and Mechanical System" Van Nostrand Renihold, 1989.

[16] P. Giacomo "A Stepping Motor Primer" Byte (vol. 4-no. 2, February 1979), (vol.4-no. 3 , March 1979).

[17] H. V Malmstadt, C. G Euke and S. R Grouch "Electronic Measurement for Scientists", WA. Benjamin 1997.

[18] Kenneth Hintz and Daniel Tabak(1976) "Microcontrollers, Architecture, Implementation and Programming" McGraw-Hill, Inc. 1992.

[19] R. L. Tokhelium "Theory and Problem of Microprocessor Fundamentals" Scharms Outline Serial, McGraw-Hill Book Corp. NewYork, 1983.

[20] Memory Components Handbook, Intel Corporation Santa Clara CA, 1986.

[21] G. B. Nebon "The Use of EPROM - based Microcomputer" WESCON'82 Conference Record, Anaheium CA Session 5C/2 Sept. 1982.

[22] L. Wheeler "The Practical EEPROM" Byte July 1983.

[23] Hawkins H.W "Concepts of Digital Electronics" Blue Summit:TAB Books, 1983.

[24] Hallmark C. "Electronics Measurements Simplified" Blue Ridge Summit:TAB Books, 1973.

[25] Tedesehi F.P "How to Design and use Electronic Control System" Blue Ridge Summit:TAB Books, 1988.

[26] Weiss D Microprocessors in Industrial Measurement and Control" Blue Ridge Summit:TAB Books, 1987.

[27] Dalglish R.L "An Introduction to control and Measurement with Microcomputers" NewYork: Cambridge university Press, 1987.

[28]Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Interchange, Standard RS 232-C, Catalog 3, Electronic Industrial Associations.

[29] J. E Olehsy and G.B Rutkowski "Microprocessor and Digital Computer Technology" Prentece Hall, 1981.

[30] J. E. McNamara "Technical Aspects of Data Communication" Digiatal press Bedford. 1977.

[31]Programming Industrial Control System Using IEC 1131-3. Published by: The Institution of Electrical Engineers, London. 1995 - from 1988. British Library Cataloguing in Publication Data. Short Run Press Ltd. Exeter. 1995.

[32] Bela G. Lipak and Kriszta Venerel (eds). "Instrument Engineer's Handbook. Publisher Chilton Book Company, Randar. 1982.

[33] Gerhard E. Hoernes and Melvin F. Helweel "Introduction to Boolean Algebra and Logic design" Library of Congress Catalog . McGraw-Hill, Inc. Publication. 1964.

[34] The complete guide to ROM-Based system software "System BIOS for IBM PCs, Compatibles and EISA Computers", Phoenix Technologies Ltd. - 2nd Edition. Addison - Wesley Publishing Company.

[35] Using Assembly Language

[36] Microsoft Visual C++ User Guides

[37] Microsoft Windows NT Resource Kits

[38] Microsoft Windows 95 Resource Kit