PLC AUTOMATION OF BEARING PRESS MACHINE

A PROJECT REPORT

Submitted by

DEEPAK.R	**412412105020**
GOKULAKRISHNAN.S	**412412105028**
JAYA CHANDRAN.S	**412412105038**
VETRIVEL.T	**412412105326**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

In

ELECTRICAL AND ELECTRONICS ENGINEERING

SRI SAI RAM INSTITUTE OF TECHNOLOGY, CHENNAI-44

ANNA UNIVERSITY: CHENNAI 600025

APRIL 2016

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**PLC AUTOMATION OF BEARING PRESS MACHINE**" is the bonafide work of the following students who carried out the project work under my supervision.

DEEPAK.R	**412412105020**
GOKULAKRISHNAN.S	**412412105028**
JAYA CHANDRAN.S	**412412105038**
VETRIVEL.T	**412412105326**

SIGNATURE
Mr.A.ANBAZHAGAN, M.Tech
HEAD OF THE DEPARTMENT
Electrical and Electronics Engineering,
Sri Sai Ram Institute of technology,
Chennai - 600 044.

SIGNATURE
Mrs.A.SASIKALA, M.Tech
SUPERVISOR
ASSISTANT PROFESSOR
Electrical and Electronics Engineering,
Sri Sai Ram Institute of technology,
Chennai - 600 044.

Submitted for ANNA UNIVERSITY Viva-Voce Examination held on …...................... at Sri Sai Ram Institute of technology, Chennai-600 044.

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGMENT

We express our deep sense of gratitude to our beloved Chairman **Thiru.MJF.Ln.LEO MUTHU** for the help and advice he has shared upon us.

We express our gratitude to our **CEO Mr.J.SAI PRAKASH LEO MUTHU** and our trustee **Mrs.J.SHARMILA RAJA** for their constant encouragement in completing the project.

We express our solemn thanks to our esteemed Principal **Dr.K.PALANIKUMAR** for having given us spontaneous and wholehearted encouragement for completing the project.

We are indebted to our HOD **Mr.A.ANBAZHAGAN** for his support during the entire course of this project work.

We express our gratitude and sincere thanks to our guide **Mrs.A.SASIKALA , Asst.Professor** for her valuable suggestions and constant encouragement for successful completion of this project.

Our sincere thanks to project coordinator **Mr. VEERASUNDARAM, Asst.Professor** for his kind support in bringing out this project successfully.

Finally, we thank all teaching and non-teaching staff members of the **DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING** and all others who contributed directly or indirectly for the successful completion of our project.

ABSTRACT

The project explains the automation of a press machine using a programmable logic controller (PLC).The operation of the machine includes fitting two bearing and a shaft within the housing. This operation should be accurate because of sensitive tolerance at the bearing and housing. The pressure applied to each machine part and its displacement within the housing is monitored using load cell and transducers like LVDT and the whole assembly process is controlled by the program logic. The machine can be operated in both automatic and manual mode. Thus the automation of this process will improve productivity, safety and assembly accuracy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER-1

INTRODUCTION

1.1 OVERVIEW OF PROJECT

The automated bearing press machine is a special purpose press machine used for assembling automobile components such as cylinder block, drive shaft, bearing and fuel injector. The automation is done by a programmable logic controller (PLC). The machine assembles two bearings and a shaft within a housing having very close tolerance values. The automation of the machine enables to accurately control the hydraulic presses so that the job is given right amount of load and displaced within the housing accurately. The transducers like LVDT are used to detect the displacement of the bearing within the housing and the load cell is used to constantly detect the load applied upon the material to be pressed. The PLC allows the programmer to make suitable interlocks in the machine to improve the safety and to provide a foolproof operation. The automated machine improves the production of the company, quality of the product delivered and also reduces the human time and cost incurred upon the machine.

1.2 EXISTING SYSTEM

The existing press machine employs APRIL-5000 model PLC, with GESPAC module. The earlier machine uses purely hydraulic cylinders for the assembly purpose. The load cell and linear variable displacement transducer. A combination of Light emitting diode and light dependent resistor is used for safety during operation of the machine.

1.2.1 DISADVANTAGES

- The APRIL-5000 PLC model does not support analog inputs directly.
- A separate GESPAC analog module was used, which needs an external card to support analog raw values.
- The future expansion is not possible because this PLC has only limited number of input and output units.
- The accuracy of assembly process reduces.

1.3 PROPOSED SYSTEM

The OMRON-PLC replaces the earlier version of APRIL-5000 PLC, which is cheaper in cost. AD041 analog module is used which supports the analog inputs directly and does not require any external card. The machine uses both hydraulic and pneumatic cylinder to increase the accuracy. A separate DC motor is provided to move the loaded bearing towards the RAM, so that process in uninterrupted and runs continuously. A pair of infrared sensor is placed inclined at the press for safety condition. The process stops, when there is a disturbance during the press. A pair of proximity sensor is placed at each fixture in the index table. As the table rotates at an angle of 45 degree, an additional interlock is provided in the PLC program to logically cut off the supply to the motor.

1.3.1 ADVANTAGES

- The cost of the PLC is less compared to the previous version.
- The analog module interfaced with this PLC supports the analog inputs directly.
- The accuracy of the assembly process increases
- Since the whole process is automated, It requires less human labour
- Maintenances cost reduces
- The future expansion is possible because a maximum of four input/output modules can be connected to the PLC.
- Time taken for completion of a job reduces.
- If there is any failure, troubleshooting is easy.

CHAPTER-2

LITERATURE SURVEY

Communications, Circuits and Systems, 2004. ICCCAS 2004. 2004 International Conference on (Volume: 2) Lei Lin; Sch. of Auto. Eng., Univ. of Electron. Sci. & Technol. of China, Chengdu, China; Houjun Wang

INFERENCE

When a PLC (programmable logic controller) and microcomputer communicate in real-time systems, the technique of real-time communication between PLC and microcomputer is used in many areas. This paper researches the technique and solves the real-time communication problem between the PLC and microcomputer. We use the OMRON PLC and C++ program language, and use the example of real-time communication between PLC and microcomputer in control of a microwave transmitter to illuminate the basic means and technique in real-time system design and communication software design, including the core technique of real-time communication. This proves the validity and the value of this technology.

- Intelligent and Advanced Systems, 2007. ICIAS 2007. International Conference

H. Hashim ; Fac. of Electr. & Electron. Eng., Univ. Tun Hussein Onn Malaysia, Batu Pahat ; Z. A. Haron

INFERENCE

Recent enhancement of the industrial communications and networking are possible to apply in Ethernet networks system at all levels of industrial automation, especially in the controller level whereby the data exchanges in real-time

communication is mandatory. This paper is about a study on the development of industrial communications network based on the Ethernet protocol and thus implements it into computer integrated manufacturing (CIM) system. The purpose of this paper is to overcome real-time communication in which the accessibility of data exchange is very difficult in terms of retrieving data from other stations and time consuming. The Ethernet module is installed on supervisory OMRONPLC to integrate several of stations in the CIM-70A system which is located at Robotic Laboratory in University Tun Hussein Onn Malaysia (UTHM) widely used in automation networking systems. It is found that, the Ethernet protocol approach through the communication and integration of CIM system can be accessed easily and available to be upgraded at the management and enterprise levels of industrial automation system.

CHAPTER-3

DDS BEARING PRESS MACHINE

3.1 WORKING

The press machine is used to assemble a primary bearing, secondary bearing and a drive shaft. The machine has an indexing table with eight fixtures which is driven by an induction motor. The machine uses both hydraulic and pneumatic cylinders for the assembly. There are three stations with RAM. Initially, housing is placed in the fixture and the cycle starts. The primary and secondary bearings will be loaded in their respective stations .After the index table rotates; drive shaft will be placed in the fixture. In the next index, housing moves to the first station and the primary bearing is assembled with correct load and position within the housing .Likewise, for one complete rotation of the table, three housings will be assembled. The drive shaft gauge is used to measure the distance between secondary bearing and the drive shaft and its tolerance is set around 40 microns.

STATION 1			
	MINIMUM	**MAXIMUM**	**THRESHOLD**
Effort 1(ton)	150	900	0.8(mm)
Position 1(mm)	123.45	123.7	1100.0(ton)
STATION 2			
Effort 2	60	1200	0.7mm
Position 2	8.8	8.94	1100(ton)
Position 3	0.035	0.5	50(ton)
STATION 3			
Effort 4	80	500	9mm
Effort 4	120	600	0.8mm
Position 4	73.755	73.845	1200(ton)

Table 3.1- Tolerance value of assembled job

3.2 HARDWARE DETAILS

3.2.1 HYDRAULIC CYLINDER

An actuation device that makes use of a pressurized hydraulic fluid is known as a hydraulic pump. This mechanism is used for producing linear motion and force in applications that transfer power. In other words, a hydraulic cylinder converts the energy stored in the hydraulic fluid into a force used to move the cylinder in a linear direction. A hydraulic cylinder consists of a cylindrical barrel, piston, and a piston rod. The piston that is placed within the barrel is connected to the piston rod. The cylinder bottom, and the cylinder head, closes the bottom and the head of the barrel respectively. The cylinder head is the side from where the piston rod exits the cylinder. The piston rod starts moving outwards, as the hydraulic fluid is pumped into the bottom side of the hydraulic cylinder. In the reverse process, the hydraulic fluid is pushed back into the reservoir by the piston. The pressure in the cylinder is the ratio of unit force per unit piston area.

Mechanical power is defined as Force x velocity. This makes it easy to calcula te the power of acylinder. The fluid power supplied is more than the mechanic al power output because of friction between the sliding parts.

$$P = F\,v \quad \text{Watts}$$

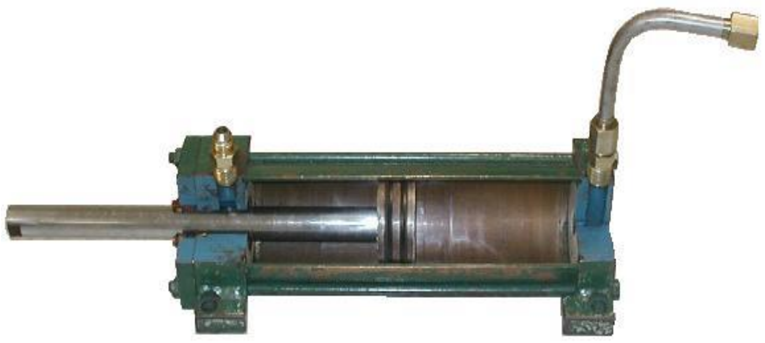


Fig 3.1- Hydraulic cylinder

3.2.2 PNEUMATIC CYLINDER

Pneumatic cylinder are mechanical devices which use the power of compressed gas to produce a force in a reciprocating linear motion.

Like hydraulic cylinders, something forces a piston to move in the desired direction. The piston is a disc or cylinder, and the piston rod transfers the force it develops to the object to be moved. Engineers sometimes prefer to use pneumatics because they are quieter, cleaner, and do not require large amounts of space for fluid storage.

Because the operating fluid is a gas, leakage from a pneumatic cylinder will not drip out and contaminate the surroundings, making pneumatics more desirable where cleanliness is a requirement.

Once actuated, compressed air enters into the tube at one end of the piston and, hence, imparts force on the piston. Consequently, the piston becomes displaced.

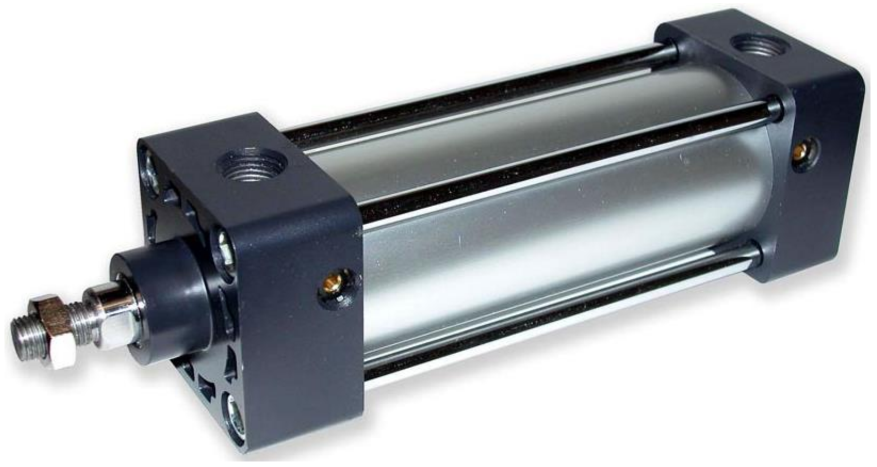


Fig 3.2- Pneumatic cylinder

3.2.3 LINEAR VARIABLE DISPLACEMENT TRANSDUCER (LVDT)

The linear variable differential transformer (LVDT) also called just a differential transformer. The LVDT converts a position or linear displacement from a mechanical reference zero, or null position into a proportional electrical signal containing phase for direction and amplitude (for distance) information. The LVDT operation does not require an electrical contact between the moving part (probe or core assembly) and the coil assembly, but instead relies on electromagnetic coupling.

The linear variable differential transformer has three solenoid coils placed end-to-end around a tube. The center coil is the primary, and the two outer coils are the top and bottom secondary. A cylindrical ferromagnetic core, attached to the object whose position is to be measured, slides along the axis of the tube. An alternating current drives the primary and causes a voltage to be induced in each secondary proportional to the length of the core linking to the secondary. The frequency is usually in the range 1 to 10 kHz.

As the core moves, the primary's linkage to the two secondary coils changes and causes the induced voltages to change. The coils are connected so that the output voltage is the difference between the top secondary voltage and the bottom secondary voltage. When the core is in its central position, equidistant between the two secondary’s, equal voltages are induced in the two secondary coils, but the two signals cancel, so the output voltage is theoretically zero. The LVDT is designed with long slender coils to make the output voltage essentially linear over displacement up to several inches (several hundred millimeters) long.

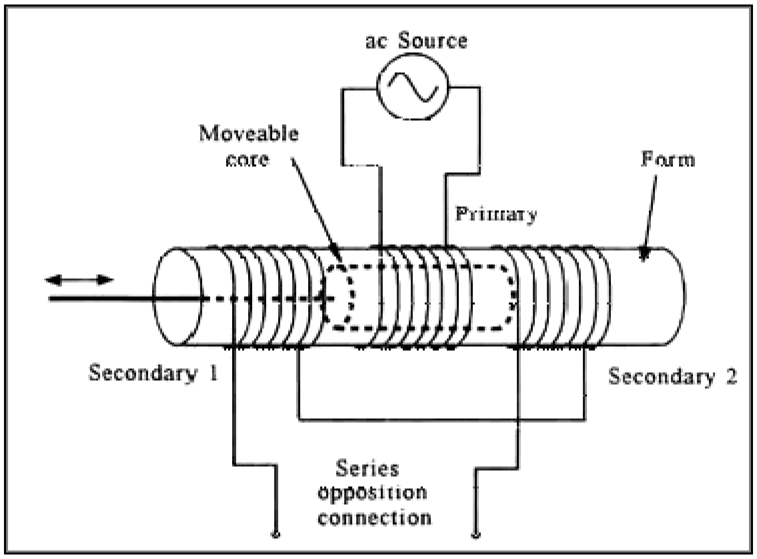


Fig 3.3- Linear variable displacement transducer

3.3 SOFTWARE DETAILS

3.3.1 HMI

HMI stands for human machine interface. This is the interface between the operator and the controller. The HMI is the controller operating panel. The panel comprises a numeric keypad and a LCD screen that displays text. The keypad is used to input data into the application, such as Timer values. The PLC Display screen can show operator messages, variable information from the program and system information. Human-machine interface is probably the sector in automation which has made the greatest progress in the last few years. This progress is due to increasingly sophisticated and user-friendly electronics and signal processing. With the right choice of interface and its configuration, users can control processes with preventive maintenance to increase productivity by reducing downtime

3.3.2 ROLE OF OPERATOR

The operating interface includes all the functions required for controlling and supervising the operation of a machine or installation. Depending on the requirements and complexity of the process, the operator may have to perform. Regular process run tasks - stop and start the process; both steps may include start and stop procedures that are automatic or manual or semi-automatic and controlled by the operator to operate the controls and make the adjustments required for regular process run and monitor its progress. Tasks to deal with unexpected events detect abnormal situations and undertake corrective action before the situation disturbs the process.

3.3.3 QUALITY OF INTERFACE DESIGN

The quality of the operating interface design can be measured by the ease with which an operator can detect and understand an event and how efficiently he can respond. Detect any change in a machine's operating conditions is usually seen by a change in or display of information on an indicator, display unit or screen. The operator must, above all, be able to detect the event in any environmental conditions (ambient lighting). Different means can be employed to attract attention, flashing information, color change, sound signal, and anti-reflection devices. For display unit clear texts in the language of the user, adequate reading distance, use of standard symbols, zoom giving a detailed view of the area the message involves, Respond Depending on what message the machine sends, the operator may have to act swiftly by pressing one or more buttons or keys. This action is facilitated by clear markings to identify buttons and keys easily, such as standard symbols on buttons.

3.3.4 PROTOCOLS SUPPORTED

Uni-TE (Uni-Telway), Modbus, Modbus TCP-IP, FipWay, Modbus Plus, and third-party protocols are also available.

3.3.5 REQUIREMENTS

The general requirements of the automotive HMIs can be summarized as follows. The HMI should inform and support the intervention considering the following aspects:

- Readability
- Clarity
- Interpretability
- Accessibility
- Ease of handling

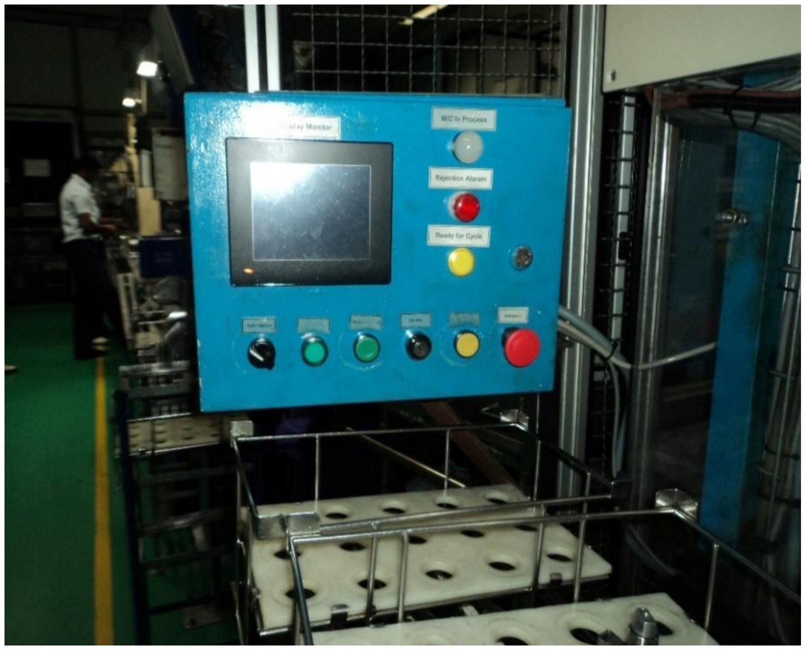


Fig 3.4 - HMI

CHAPTER-4

PROTOTYPE MODEL

4.1 BLOCK DIAGRAM

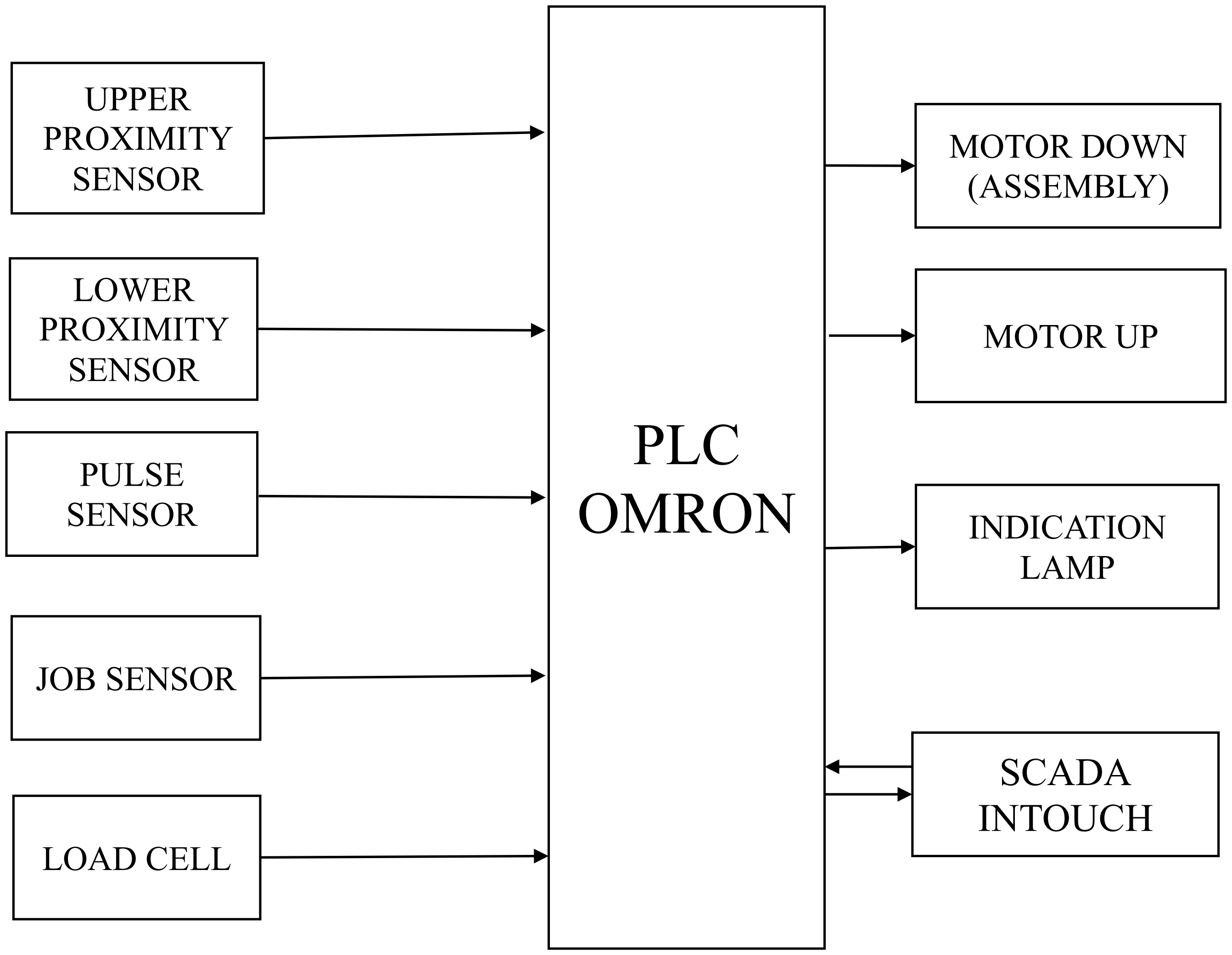


Fig 4.1- block diagram of prototype

4.2 WORKING

The prototype works in two basic modes: automatic and manual mode. The main objective is to assemble the job inside the housing at the right position by applying the right pressure. In the automatic mode, total human time required is minimal, the operator needs only to place the job in the fixture and start the cycle. The PLC checks for the home condition and starts the assembly cycle by rotating the motor down. The load applied onto the job and the pulse given to the motor is constantly monitored. The written logic in the PLC controller compares these values with the pre-set value and stops the motor when the job is displaced the right amount within the housing and enables the motor up logic to return the motor to the home condition. The indication for the correctly assembled product is provided using tower lamps. The manual mode is provided as a backup when the automatic mode fails. Since, it works without the interference of PLC, the operator needs to ensure the tolerance of the assembled job. Interlocks are provided in the logic to ensure fool-proof operation and safety. The operator is informed when job is assembled wrongly using fault alarm.

4.2.1 SEQUENCE OF OPERATION

- Job presence sensor ON, up limit proximity sensor ON, Auto switch ON.
- Yellow PB and Blue PB ON.
- Cycle starts.
- Motor rotates down (job pressed), upper limit proximity sensor OFF.
- Pulse sensor ON, Load cell ON, lower limit proximity sensor ON.
- Infrared Sensor ON
- Motor rotates up, lower limit proximity sensor OFF.
- Cycle complete.
- Emergency PB ON, cycle interrupts.

4.3 HARDWARE AND SOFTWARE DETAILS

4.3.1 LOAD CELL

A load cell is a transducer that is used to create an electrical signal whose magnitude is directly proportional to the force being measured. The various types of load cells include hydraulic load cells, pneumatic load cells and strain gauge load cells.

4.3.1.1 STRAIN GAUGE LOAD CELL:

The strain gauge measures the deformation (strain) as a change in electrical resistance, which is a measure of the strain and hence the applied forces. A load cell usually consists of four strain gauges in a Wheatstone bridge configuration. Load cells of one strain gauge (Quarter Bridge) or two strain gauges (half bridge) are also available. The electrical signal output is typically in the order of a few millivolts and requires amplification by an instrumentation amplifier before it can be used. The output of the transducer can be scaled to calculate the force applied to the transducer. Sometimes a high resolution ADC, typically 24-bit, can be used directly.

These load cells are particularly stiff, have very good resonance values, and tend to have long life cycles in application. Strain gauge load cells work on the principle that the strain gauge (a planar resistor) deforms/stretches/contracts when the material of the load cells deforms appropriately. These values are extremely small and are relational to the stress and/or strain that the material load cell is undergoing at the time. The change in resistance of the strain gauge provides an electrical value change that is calibrated to the load placed on the load cell.

Strain gauge load cells convert the load acting on them into electrical signals. The gauges themselves are bonded onto a beam or structural member that deforms when weight is applied. In most cases, four strain gauges are used to obtain maximum sensitivity and temperature compensation. Two of the gauges are usually in tension, and two in compression, and are wired with compensation adjustments. The strain gauge load cell is fundamentally a spring optimized for strain measurement. Gauges are mounted in areas that exhibit strain in compression or tension. The gauges are mounted in a differential bridge to enhance measurement accuracy. When weight is applied, the strain changes the electrical resistance of the gauges in proportion to the load. Other load cells are fading into obscurity, as strain gauge load cells continue to increase their accuracy and lower their unit costs.

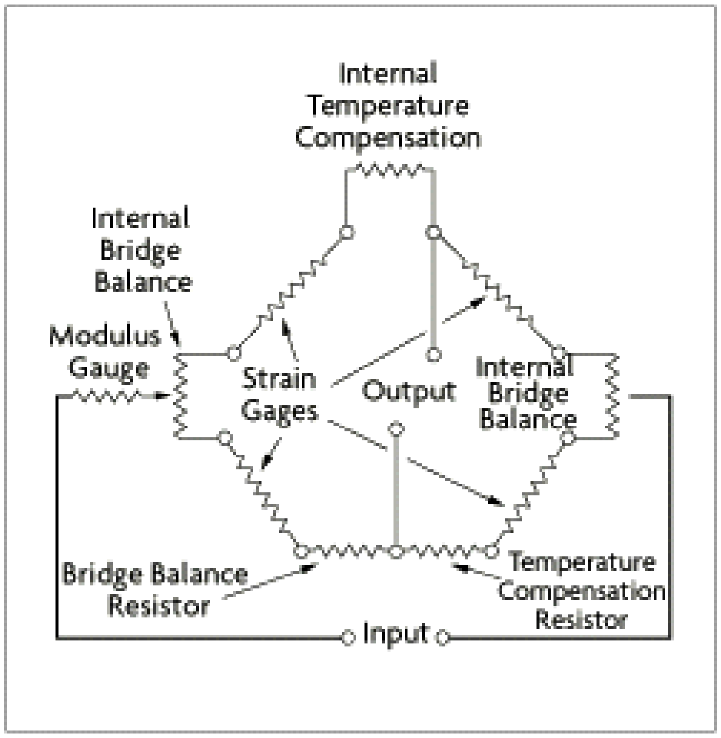


Fig 4.2- strain gauge load cell

4.3.2 PROXIMITY SENSOR

A proximity sensor is a sensor able to detect the presence of nearby objects without any physical contact. Proximity sensors can have a high reliability and long functional life because of the absence of mechanical parts and lack of physical contact between sensor and the sensed object.

4.3.2.1 INDUCTIVE PROXIMITY SENSOR

Inductive proximity sensors are used for non-contact detection of metallic objects. Their operating principle is based on a coil and oscillator that creates an electromagnetic field in the close surroundings of the sensing surface. The presence of a metallic object (actuator) in the operating area causes a dampening of the oscillation amplitude. The rise or fall of such oscillation is identified by a threshold circuit that changes the output of the sensor. The operating distance of the sensor depends on the actuator’s shape and size and is strictly linked to the nature of the material.

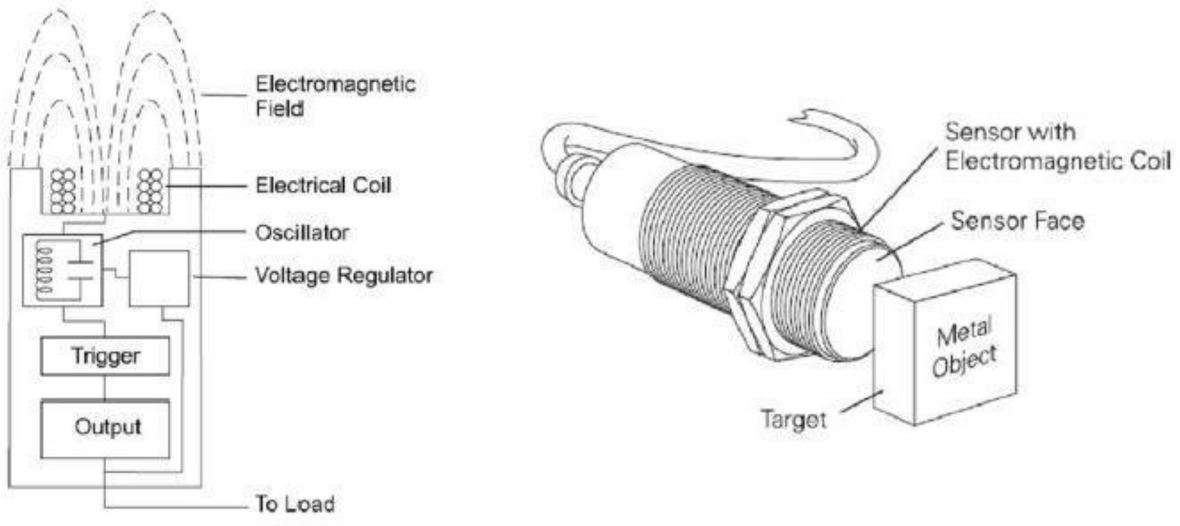


Fig 4.3 - Inductive proximity sensor

4.3.3 LIMIT SWITCH

In its simplest form, a limit switch is a “switch” that can be mounted into remote locations so that it is actuated by an object other than a human operator. Some basic functions of limit switches are:

• Detecting presence/absence

• Counting

• Detecting range of movement

• Detecting positioning & travel limit

• Breaking a live circuit when unsafe conditions arise

As the object makes contact with the actuator of the switch, it eventually moves the actuator to its “limit” where the contacts change state. A limit switch is an electromechanical device that consists of an actuator mechanically linked to a set of contacts. When an object comes into contact with the actuator, the device operates the contacts to make or break an electrical connection. Limit switches are used in a variety of applications and environments because of their ruggedness, ease of installation, and reliability of operation.

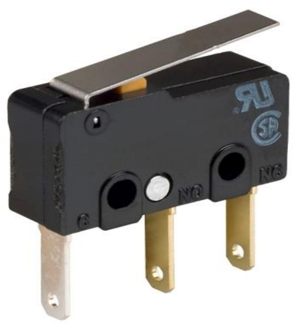


Fig 4.4- Limit switch

4.3.4 RELAY

It is an electrically worked switch. Various exchanges use an electromagnet to mechanically work a switch, yet other working models are similarly used, for instance, solid state exchanges. Exchanges are used where it is vital to control a circuit by a low-power signal (with complete electrical withdrawal amidst control and controlled circuits), or where a couple of circuits must be controlled by one sign. The main transfers were utilized as a part of long separation broadcast circuits as enhancers: they rehashed the sign rolling in from one circuit and re-transmitted it on another circuit.

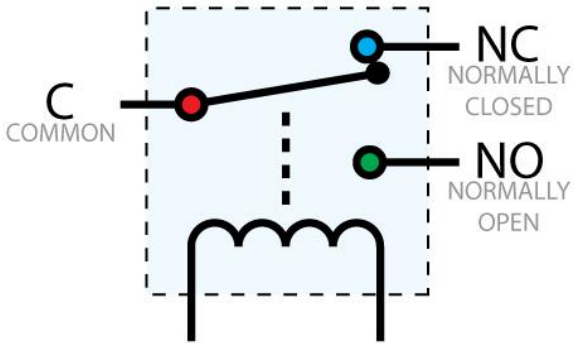


Fig 4.5 - Relay

4.3.5 TRANSFORMER

Transformers are rated in Volt-amperes (VA), or in larger units of Kilo Volt Amperes (Kva). 12-0-12 means that the voltage or the potential difference between each of the end terminals of the secondary winding and the mid-point of the secondary winding of the Transformer is 12V and between the two ends of the secondary winding, we will get 12 + 12 = 24V. 500Ma means the current delivery capability of the secondary winding of the Transformer. Normally it is said in VA. It would be 25 x 0.5 = 12VA. The ratings are based on the requirements of the loads that are to be connected to the Transformer.

Fig 4.6 - Transformer

4.3.6 VOLTAGE REGULATOR (LM7805, LM7812)

A LM7805 Voltage Regulator is a voltage controller that yields +5 volts. A simple approach to recollect the voltage yield by a LM78XX arrangement of voltage controllers is the last two digits. It yields 5 volts. The “78” section is simply the tradition that the chip creators utilization to indicate the arrangement of controllers that yield positive voltage.

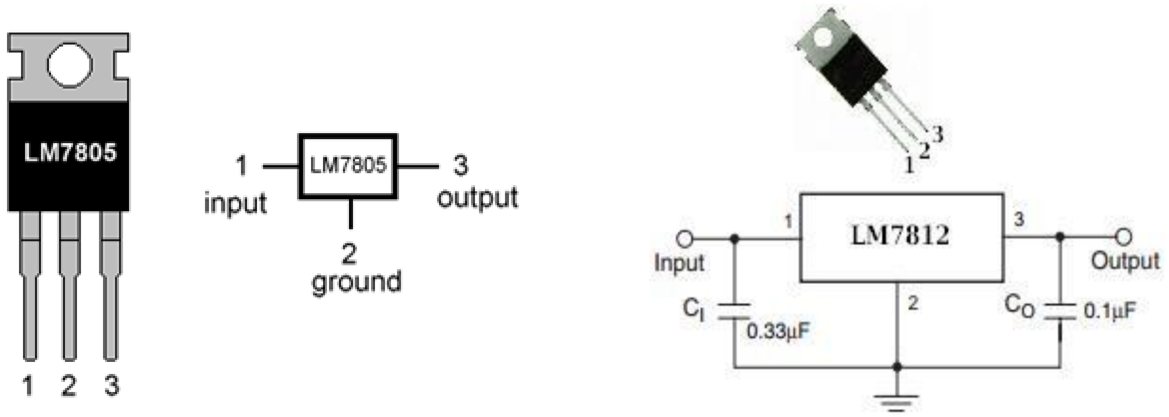


Fig 4.7 - Voltage regulator

4.3.7 DC GEAR MOTOR

A gear motor is a specific type of electrical motor that is designed to produce high torque while maintaining a low horsepower, or low speed, motor output. Gear motors are primarily used to reduce speed in a series of gears, which in turn creates more torque. This is accomplished by an integrated series of gears or a gear box being attached to the main motor rotor and shaft via a second reduction shaft. The second shaft is then connected to the series of gears or gearbox to create what is known as a series of reduction gears. Generally speaking, the longer the train of reduction gears, the lower the output of the end, or final, gear will be. Gear motors are commonly used in commercial applications where a piece of equipment needs to be able to exert a high amount of force in order to move a very heavy object.

A worm drive is a gear arrangement in which a worm (which is a gear in the form of a screw) meshes with a worm gear (which is similar in appearance to a spur gear). The two elements are also called the worm screw and worm wheel. A gearbox designed using a worm and worm-wheel is considerably smaller than one made from plain spur gears, and has its drive axes at 90° to each other.

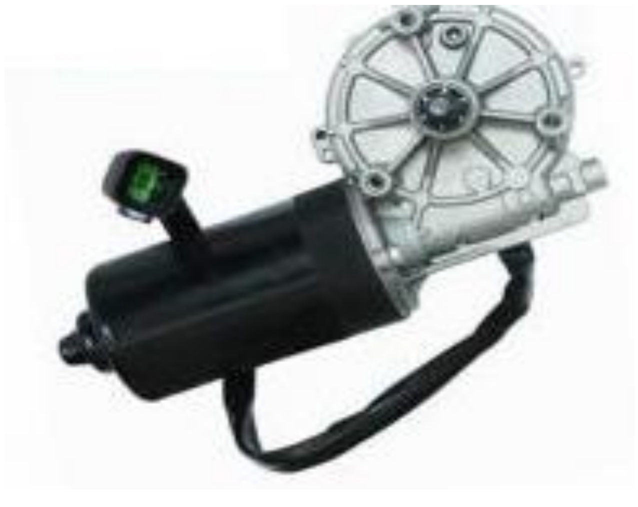


Fig 4.8 - DC gear motor

4.4 OVERVIEW OF PLC

A programmable logic controller, PLC, or programmable controller is a digital computer usedfor automation ofTypicallyindustrial electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or light fixtures. PLCs are used in many machines, in many industries.

PLCs are designed for multiple arrangements of digital and analog inputs and outputs, extended temperature ranges, immunity to electrical noise, and resistance to vibration and impact. Programs to control machine operation are typically stored in battery-backed-up or non-volatile memory.

Before the PLC, control, sequencing, and safety interlock logic for manufacturing automobiles was mainly composed of relays, cam timers, drum sequencers, and dedicated closed-loop controllers.

Since these could number in the hundreds or even thousands, the process for updating such facilities for the yearly model change-over was very time consuming and expensive, as electricians needed to individually rewire the relays to change their operational characteristics.

An industrial control computer would have several attributes: it would tolerate the shop-floor environment, it would support discrete (bit-form) input and output in an easily extensible manner, it would not require years of training to use, and it would permit its operation to be monitored.

The response time of any computer system must be fast enough to be useful for control; the required speed varying according to the nature of the process.

Early PLCs were designed to replace relay logic systems. These PLCs were programmed in ladder logic which strongly resembles a schematic diagram of relay logic. This program notation was chosen to reduce training demands for the existing technicians. Modern PLCs can be programmed in a variety of ways, from the relay-derived ladder logic to programming languages such as specially adapted dialects of BASIC and C.

Another method is state logic a very high-level programming language designed to program PLCs based on state transition diagrams

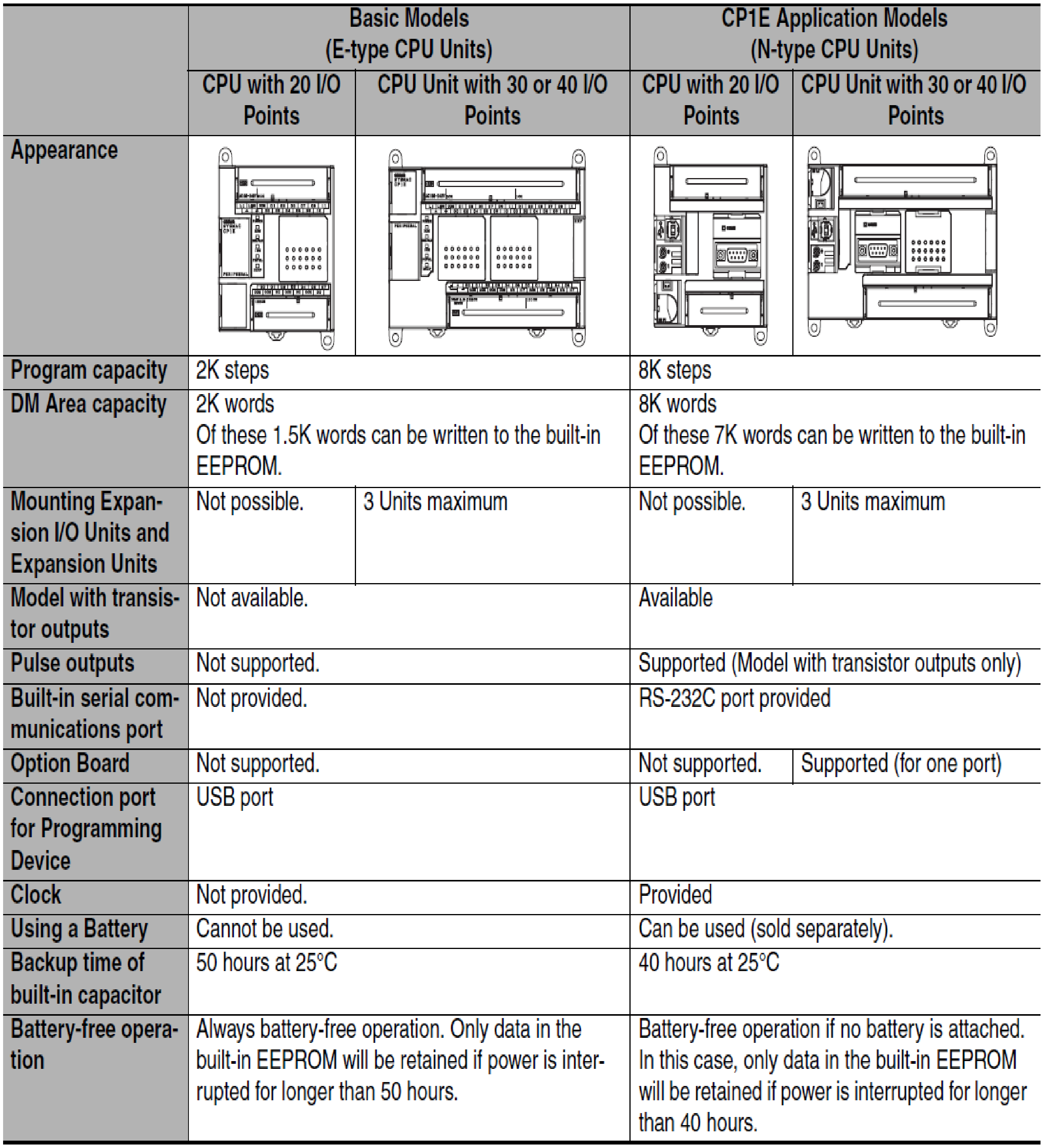
4.4.1 OMRON PLC

The SYSMAC CP1E Programmable Controller is a package-type PLC made by OMRON that is designed for easy application. The CP1E includes E-type CPU Units (basic models) for standard control operations using basic, movement, arithmetic, and comparison instructions, and N-type CPU Units (application models) that supports connections to Programmable Terminals, Inverters, and Servo Drives.

• Easy ladder programming with "smart input"

• Full motion control instruction for 2 × 100 kHz pulse output and 4/6 high-speed counter for close loop control

• Optional RS-232C or RS-422/485 communication port



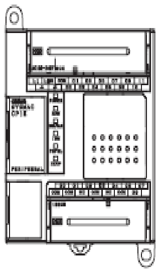
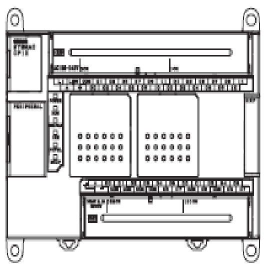
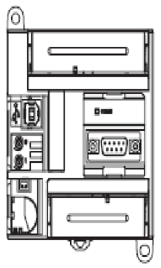
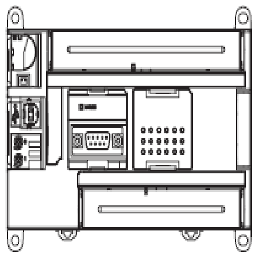
	Basic Models (E-type CPU Units)		CP1E Application Models (N-type CPU Units)	
	CPU with 20 I/O Points	CPU Unit with 30 or 40 I/O Points	CPU with 20 I/O Points	CPU Unit with 30 or 40 I/O Points
Appearance				
Program capacity	2K steps		8K steps	
DM Area capacity	2K words Of these 1.5K words can be written to the built-in EEPROM.		8K words Of these 7K words can be written to the built-in EEPROM.	
Mounting Expansion I/O Units and Expansion Units	Not possible.	3 Units maximum	Not possible.	3 Units maximum
Model with transistor outputs	Not available.		Available	
Pulse outputs	Not supported.		Supported (Model with transistor outputs only)	
Built-in serial communications port	Not provided.		RS-232C port provided	
Option Board	Not supported.		Not supported.	Supported (for one port)
Connection port for Programming Device	USB port		USB port	
Clock	Not provided.		Provided	
Using a Battery	Cannot be used.		Can be used (sold separately).	
Backup time of built-in capacitor	50 hours at 25°C		40 hours at 25°C	
Battery-free operation	Always battery-free operation. Only data in the built-in EEPROM will be retained if power is interrupted for longer than 50 hours.		Battery-free operation if no battery is attached. In this case, only data in the built-in EEPROM will be retained if power is interrupted for longer than 40 hours.	

Table 4.1 - Basic model Vs. Application model

4.4.2 CPU UNIT MODEL

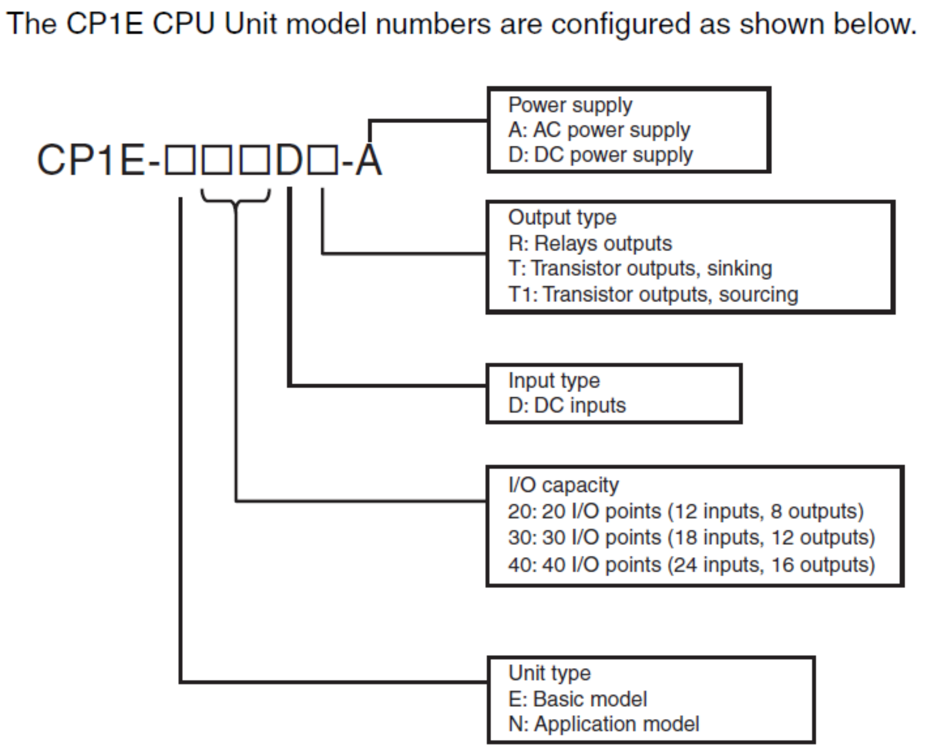

The CP1E CPU Unit model numbers are configured as shown below.
CP1E-□□□D□-A
Power supply
A: AC power supply
D: DC power supply
Output type
R: Relays outputs
T: Transistor outputs, sinking
T1: Transistor outputs, sourcing
Input type
D: DC inputs
I/O capacity
20: 20 I/O points (12 inputs, 8 outputs)
30: 30 I/O points (18 inputs, 12 outputs)
40: 40 I/O points (24 inputs, 16 outputs)
Unit type
E: Basic model
N: Application model

Fig 4.9 - CPU unit model

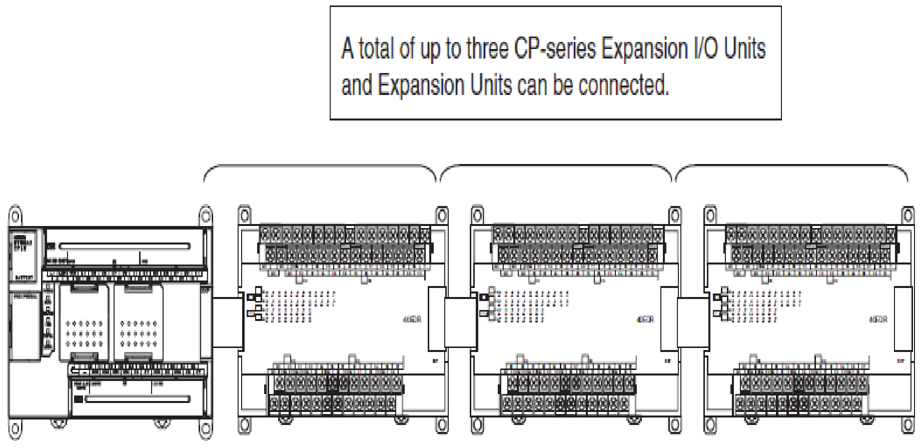

A total of up to three CP-series Expansion I/O Units and Expansion Units can be connected.

Fig 4.10 - CPU with IO units

4.4.3 PROGRAMMING SOFTWARE APPLICATION

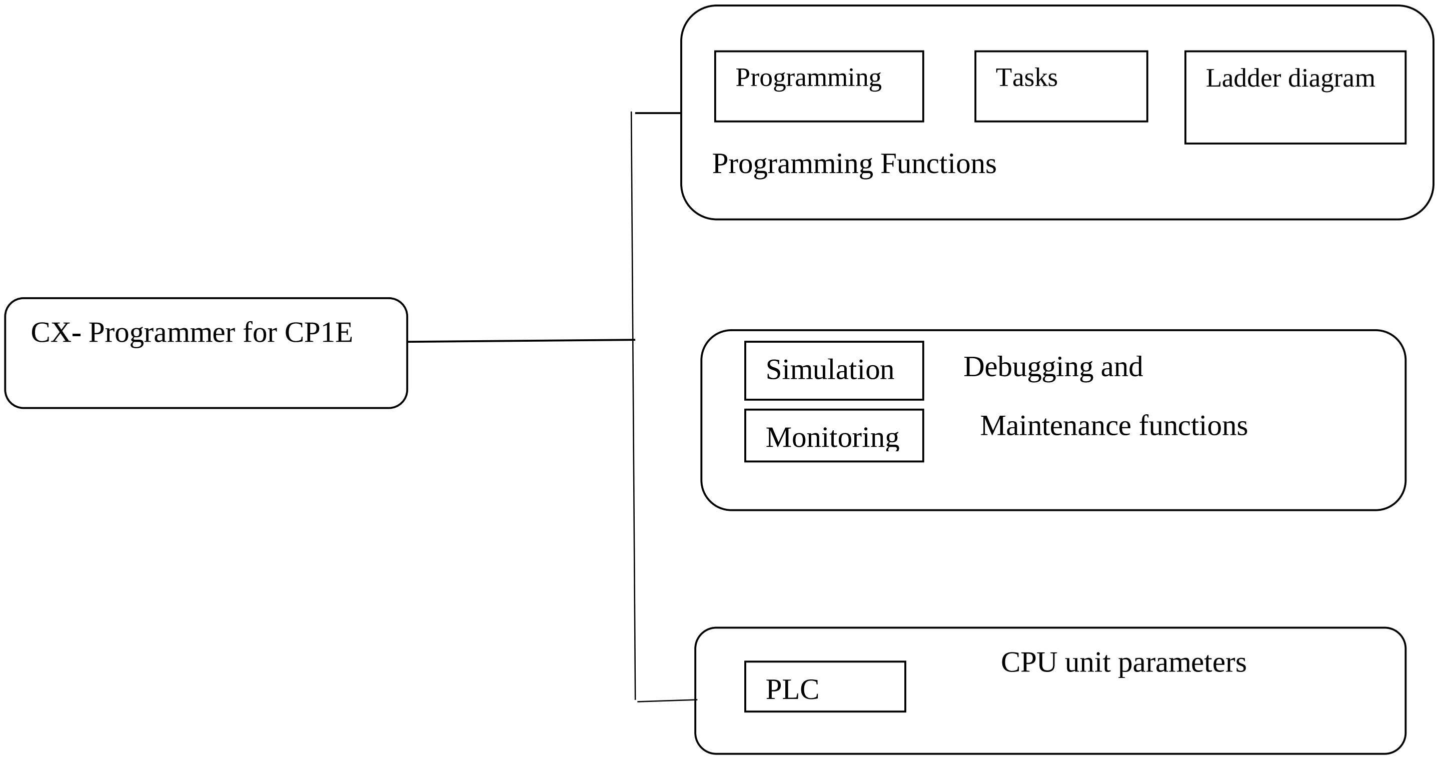
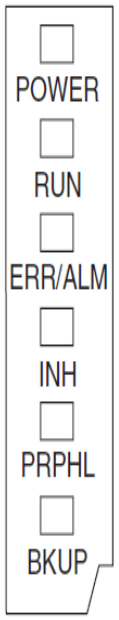
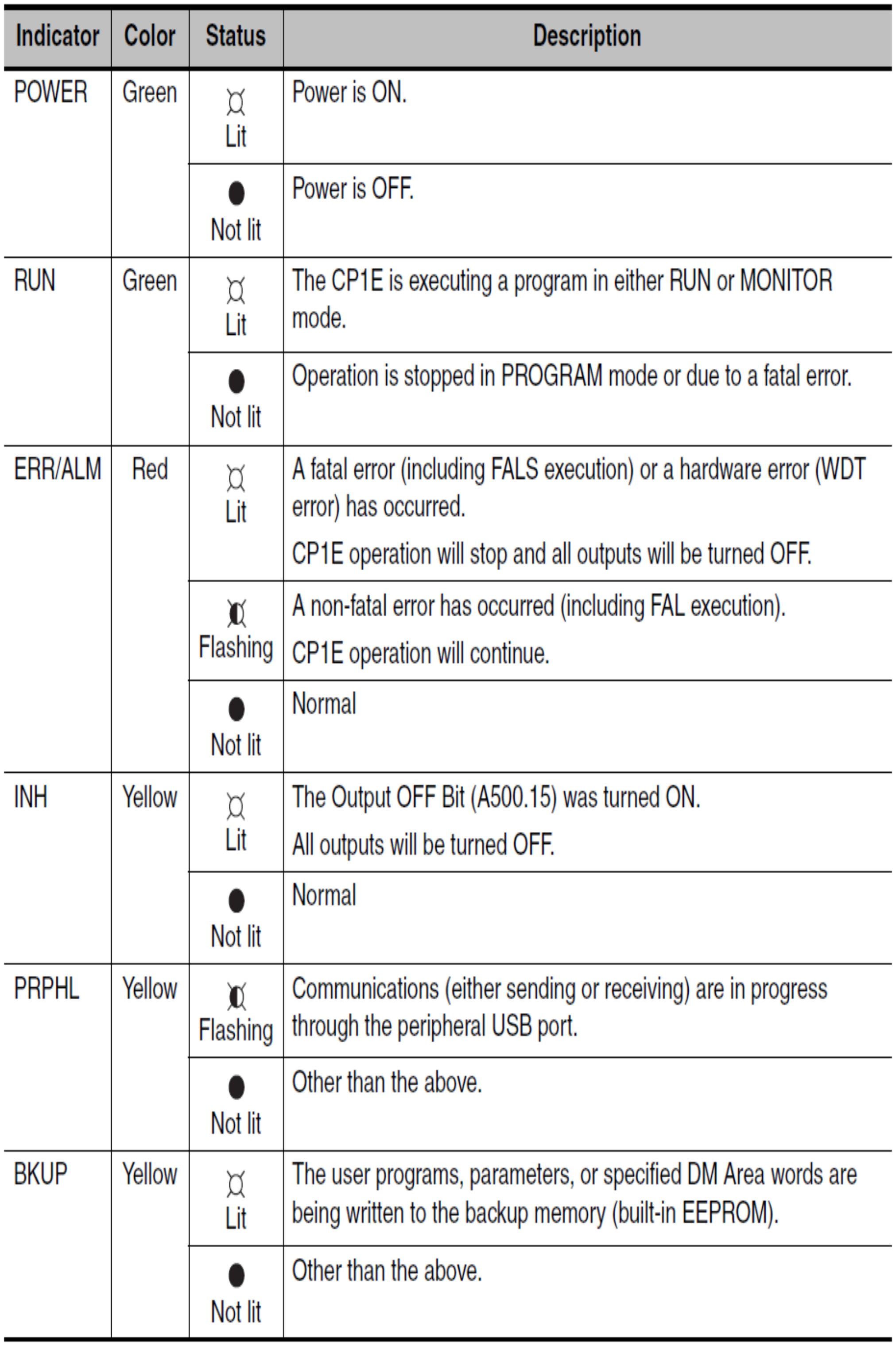


Fig 4.11 - CX programming software

The CX-Programmer for CP1E is a basic software application for creating and debugging PLC pro-grams. The programming software is used for creating a program using a ladder diagram, simulation, monitoring, troubleshooting.

4.4.4 CPU STATUS INDICATORS





Indicator	Color	Status	Description
POWER	Green	☼ Lit	Power is ON.
		● Not lit	Power is OFF.
RUN	Green	☼ Lit	The CP1E is executing a program in either RUN or MONITOR mode.
		● Not lit	Operation is stopped in PROGRAM mode or due to a fatal error.
ERR/ALM	Red	☼ Lit	A fatal error (including FALS execution) or a hardware error (WDT error) has occurred. CP1E operation will stop and all outputs will be turned OFF.
		◐ Flashing	A non-fatal error has occurred (including FAL execution). CP1E operation will continue.
		● Not lit	Normal
INH	Yellow	☼ Lit	The Output OFF Bit (A500.15) was turned ON. All outputs will be turned OFF.
		● Not lit	Normal
PRPHL	Yellow	◐ Flashing	Communications (either sending or receiving) are in progress through the peripheral USB port.
		● Not lit	Other than the above.
BKUP	Yellow	☼ Lit	The user programs, parameters, or specified DM Area words are being written to the backup memory (built-in EEPROM).
		● Not lit	Other than the above.

Fig 4.12 - CPU status indicator

4.4.5 BASIC SYSTEM CONFIGURATION USING A N- TYPE CPU UNIT

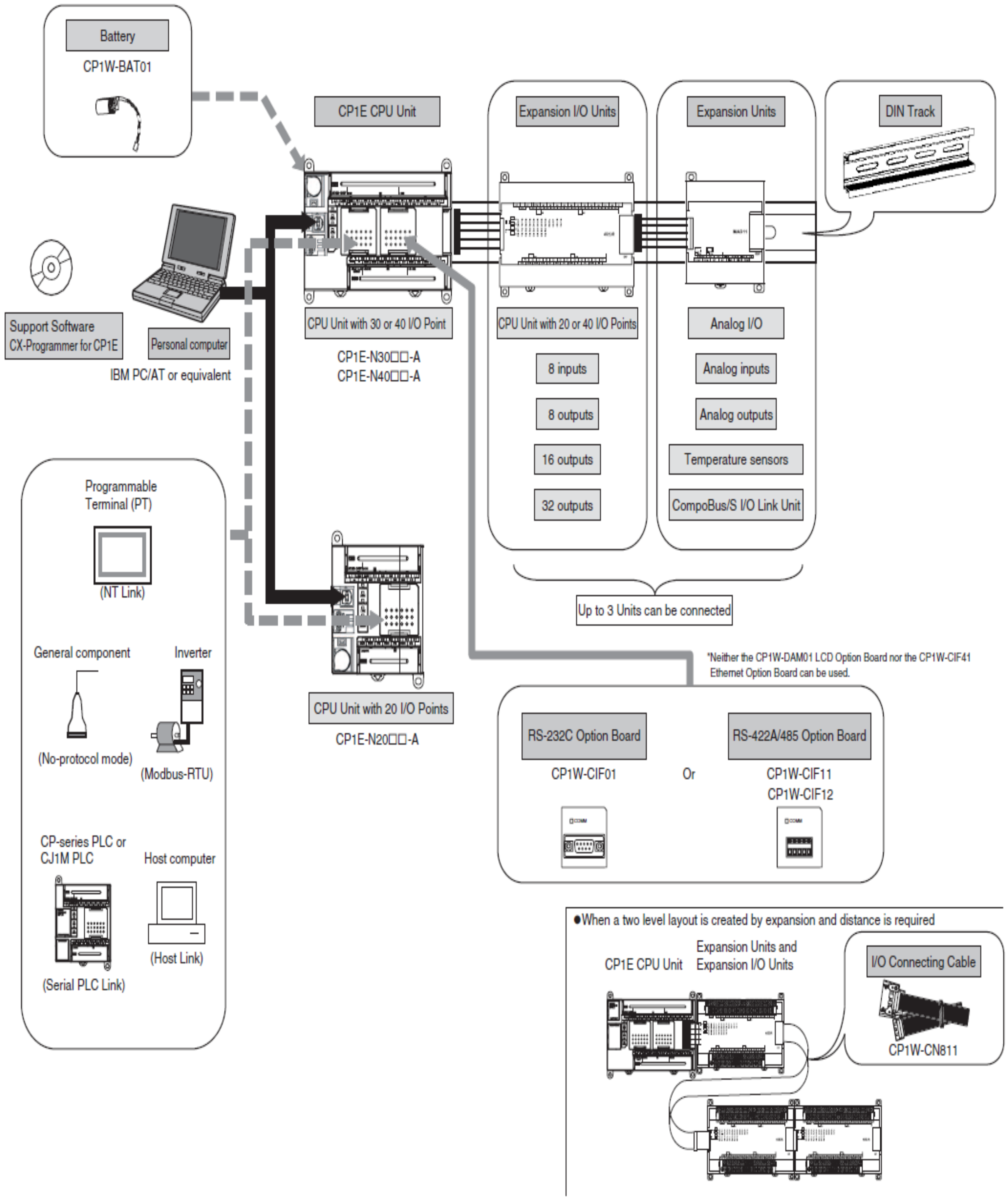


Fig 4.13 - System configuration

4.4.6 RELAY WIRING FOR MOTOR ROTATION

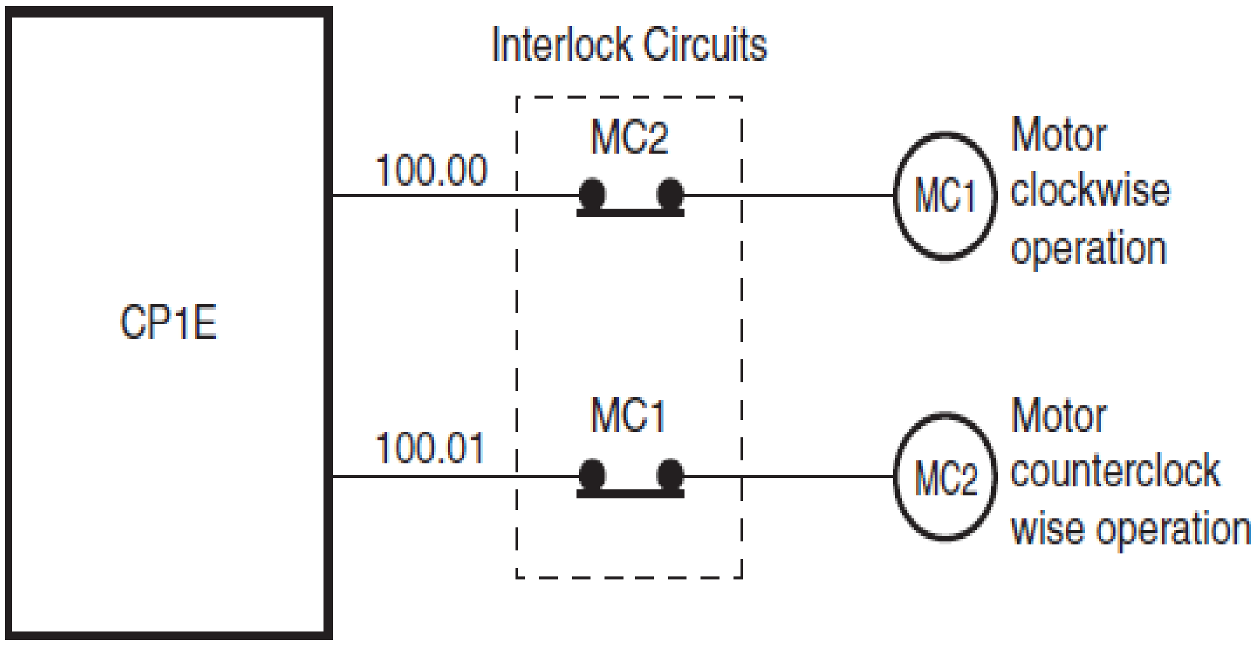


Fig 4.14 - Relay wiring

When the PLC controls operation such as the clockwise and counterclockwise operation of a motor and if there is any possibility of an accident or mechanical damage due to faulty PLC operation, provide an external interlock such as the one shown below to prevent both the forward and reverse outputs from turning ON at the same time by adding a normally closed contact MC2 during motor clock wise operation and connecting MC1 during motor reverse operation.

4.4.7 ANALOG INPUT/OUTPUT RANGE

Each CP1W-AD041 Analog Input Unit provides four analog inputs. The analog input signal ranges are 0 to 5 V, 1 to 5 V, 0 to 10 V, -10 to +10 V, 0 to 20 mA, and 4 to 20 mA. The resolution is 1/6,000.

The open-circuit detection function is activated in the ranges of 1 to 5 V and 4 to 20 mA. The Analog Input Unit uses four input words and two output words.

4.4.8 AVERAGING FUNCTION

For analog inputs, the averaging function operates when the averaging bit is set to 1. The averaging function outputs the average (a moving average) of the last eight input values as the converted value. This function is used to smooth inputs that vary at a short interval.

4.4.9 WRITING SET DATA

Write the settings for input use, averaging use, and range codes for words n+1 and n+2. When the set data is transferred from the CPU Unit to the Analog Input Unit, the A/D con-version will be started

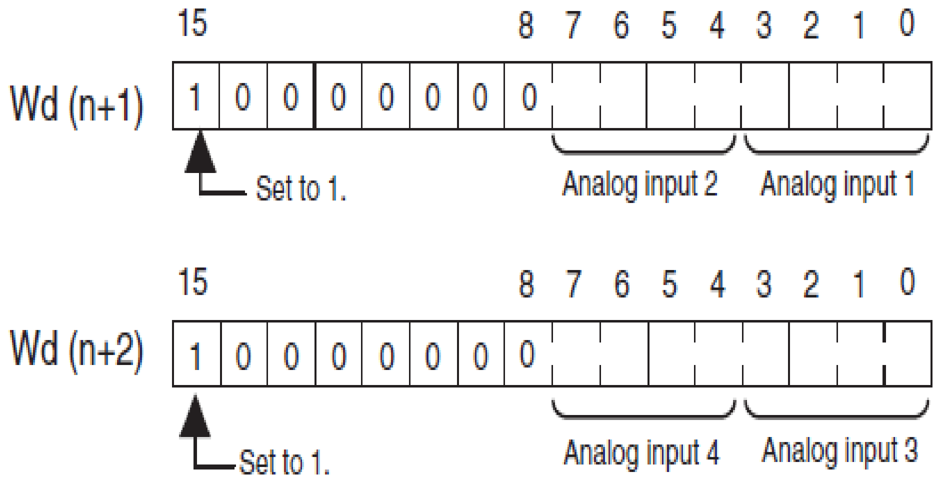


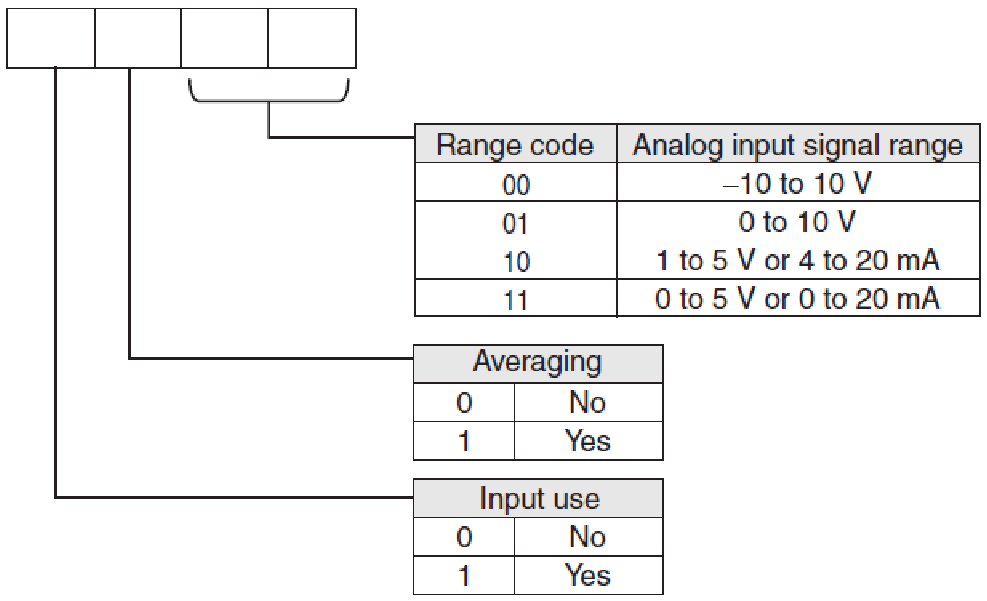
Fig 4.15 - Writing set data

The Analog Input Unit will not start converting analog I/O values until the range code has been written.

The conversion data will be 0000 until the range code is written.

Once the settings have been made, it is not possible to change the settings while power is being supplied to the CPU Unit. To change the range code or other settings, turn the CPU Unit OFF then ON again.

- Set Data



Range code	Analog input signal range
00	–10 to 10 V
01	0 to 10 V
10	1 to 5 V or 4 to 20 mA
11	0 to 5 V or 0 to 20 mA

Averaging	
0	No
1	Yes

Input use	
0	No
1	Yes

Fig 4.16 - Set data with range code

4.4.10 READING ANALOG INPUT CONVERSION VALUES

The ladder program can be used to read the memory area words where the converted values are stored. With word m as the last input word allocated to the CPU Unit or an already-connected Expansion Unit, the A/D conversion data will be output to the following words m+1 to m+4.Reading Range Code Settings and A/D Conversion Data

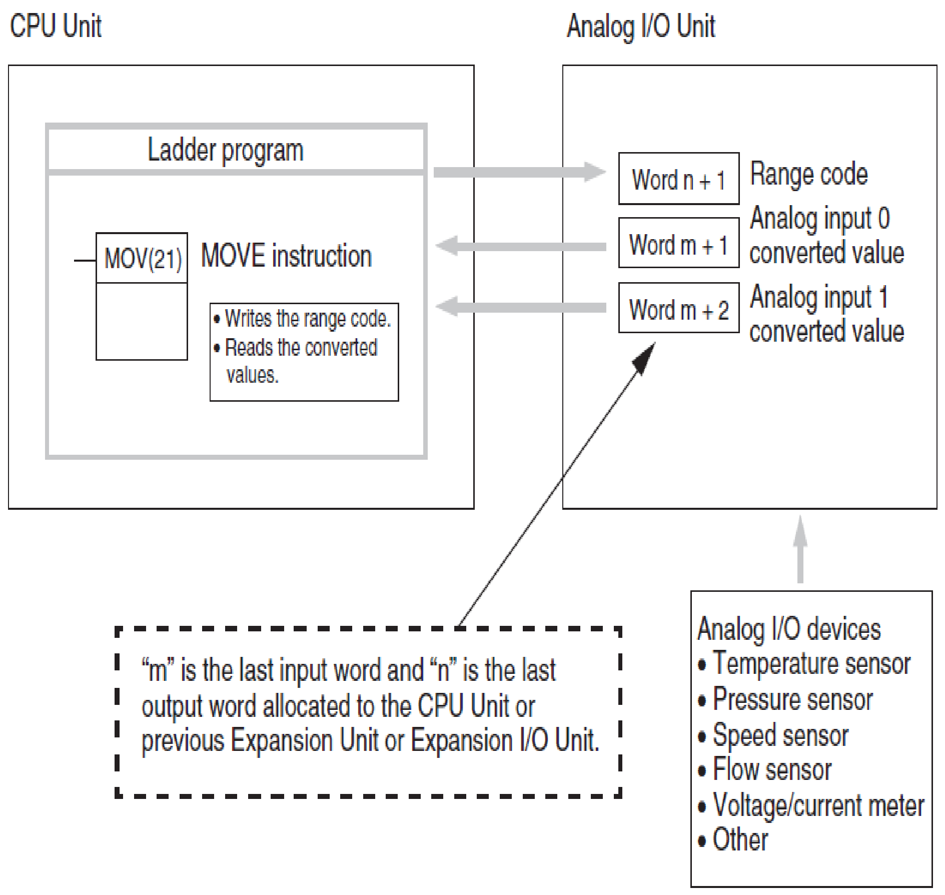


Fig 4.17 - CPU with analog unit

4.5 SOFTWARE DETAILS

4.5.1 SCADA (INTOUCH VER 10.1)

SCADA (supervisory Control and Data Acquisition) is a system for remote monitoring and control that operates with coded signals over communication channels (using typically one communication channel per remote station).

The control system may be combined with a data acquisition system by adding the use of coded signals over communication channels to acquire information about the status of the remote equipment for display or for recording functions.[1] It is a type of industrial control system (ICS).

Industrial control systems are computer-based systems that monitor and control industrial processes that exist in the physical world. SCADA systems historically distinguish themselves from other ICS systems by being large-scale processes that can include multiple sites, and large distances

4.5.2 SMART SYMOBLS:

Smart Symbols are Wonder ware In Touch graphics that are converted into reusable templates. Changes that you make to a Smart Symbol template propagate to all instances of the Smart Symbol throughout the application. This relieves you of the duplicate effort for creating, modifying, validating, and re-validating graphics used repetitively throughout an application.

The Smart Symbol Manager to import, exports, and organize the contents of the Smart Symbol library. Using Window Maker, you can create Smart Symbol templates and instances. You create Smart Symbol templates by drawing one or more graphics in Window Maker, combining them into a cell, and then converting the cell into a Smart Symbol. You can use an InTouchView application as the visual interface for applications designed specifically for a Wonder ware Application Server environment.

InTouchView applications run in Window Viewer with Application Server providing most of the SCADA functionality.

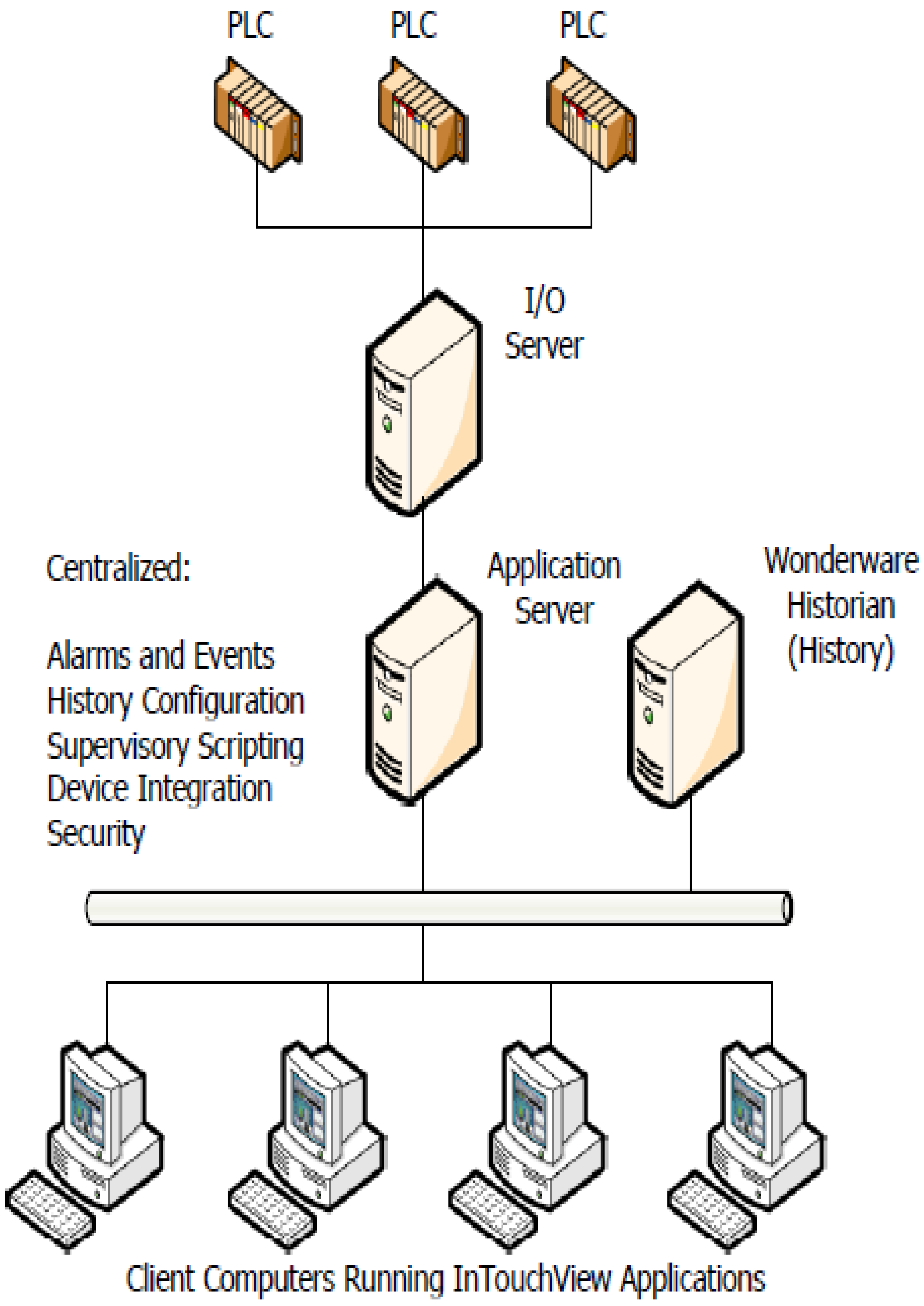


Fig 4.18 - SCADA

4.5.3 VISUALIZATION-WINDOWS AND GRAPHICS

The Window Maker to create the visual interface used by operators to view and manage your manufacturing processes. An In Touch interface shows data from and writes data back to the production environment. You can configure the following visual interface elements of your In Touch applications with Window Maker: Windows are panels containing visual elements for plant operators to manage a production process. Basic objects are simple graphical elements, such as rectangles, circles, lines, and text. User-defined complex objects consist of one or more basic objects that represent elements in your production environment, such as valves and tanks.Pre-defined complex objects perform specific functions, such as alarming and historical trending. Animation links are properties of simple and complex objects to animate their appearance and transmit user input to tasks and production data changes. Wizards are pre-defined complex objects that perform specific functions or have specific appearance, such as sliders and meters.

4.5.4 WIZARDS

Wizards are pre-designed, pre-built, and pre-programmed objects you only need to select, place, and configure for your application. Using wizards, you do not spend time drawing the individual components for the object, entering the value ranges for the object, or animating the object. Each wizard includes a configuration dialog box. You determine the behavior of your wizard instance by setting its properties the dialog box.

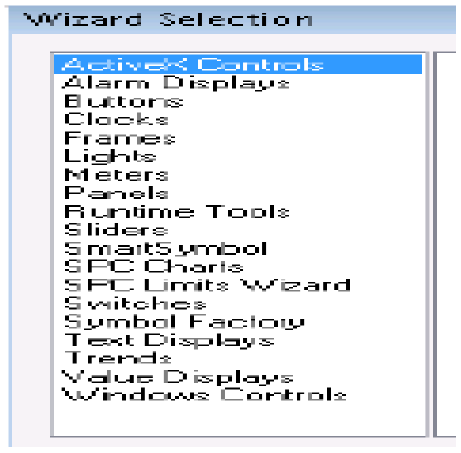


Fig 4.19 - Wizard selection

4.5.5 ANIMATION LINKS

Animation links are properties of objects that appear in InTouch windows. You can set the behavior and appearance of objects by configuring their animation links. Visualization animations modify the visual characteristics of an object like color, fill, line style, text fonts, or blinking. Interaction animations specify how users can interact with a visual element like a push button or a slider. As the user interacts with an object, the animation link alters the value of the tag linked to the object.

4.5.6 INTERACTING WITH ANIMATION LINKS

Display links show information to operators. Examples of display links are window elements changing colors, changing fill levels, moving vertically or horizontally, and blinking.

Interaction links allow operators to interact with a window element to monitor or manage an application. Examples of interaction links are sliders or push buttons that respond to operator input.You can animate color changes to any object by using color links. Changes can be based on the value of an analog or discrete tag, the value of an analog or discrete expression, or a discrete or analog alarm status.

You can use three kinds of color links to animate objects:

- Line Color
- Fill Color
- Text Color

TO CREATE A DISCRETE FILL COLOUR LINK

1Right-click the object and click Animation Links. The Animation Links dialog box appears.

2 In the Fill Color area click Discrete. The Fill Color ->Discrete Expression dialog box appears.

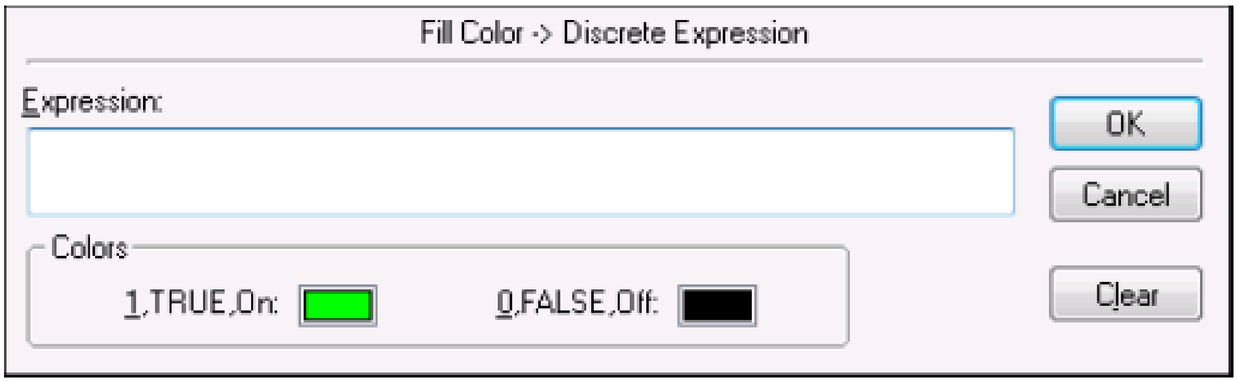


Fig 4.20 - Discreet expression dialog box

3.In the Expression box, type the name of a discrete tag or discrete expression that equates to true or false.

4 In the Colors area, click each color box to open the color palette. Select the color to use for each state.

5 Click OK.

4.5.7 ENABLING VISIBILITY

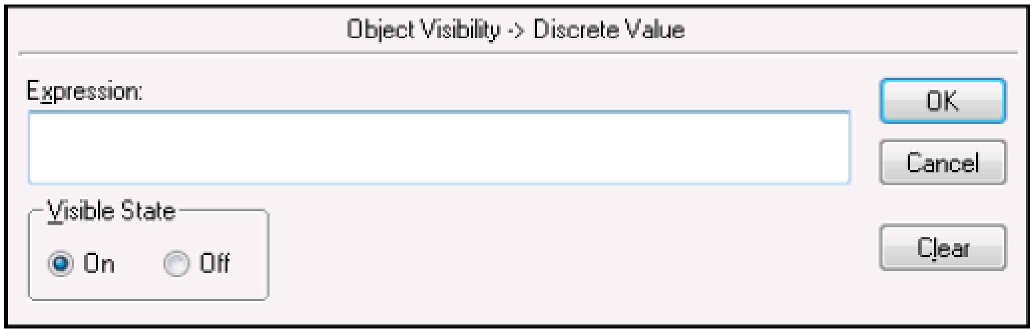
You can create links to hide objects based on the values of various tags by using visibility links. Using visibility links, you can Create the impression that moving objects only move in one direction, by hiding them when they move in the wrong direction.

- Create the impression that a moving object has stopped.
- Cause an object such as an alarm or error message to become visible only when it is activated.

To create a visibility link

1 Right-click the object and then select Animation Links. The Animation Links dialog box appears.

2 In the Miscellaneous area click Visibility. The Object Visibility Discrete Value dialog box appears.



Visible State: [✓ On] Off

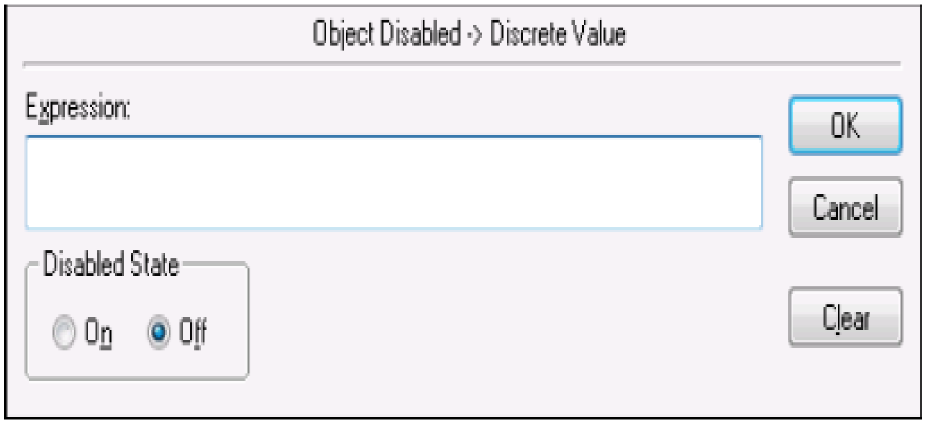
Fig 4.21- Discrete tag value dialog box

3 In the Expression box, type the name of a discrete tag or an expression that equates to a discrete value.

4 Select the Visible State for the object. If you select off, the object is invisible when the value of the expression is true. If you select on, the object is visible when the value of the expression is true.

5 Click OK.

4.5.8 DISABLE OBJECTS



Disabled State: On [✓ Off]

Fig 4.22 - Disabling object dialog box

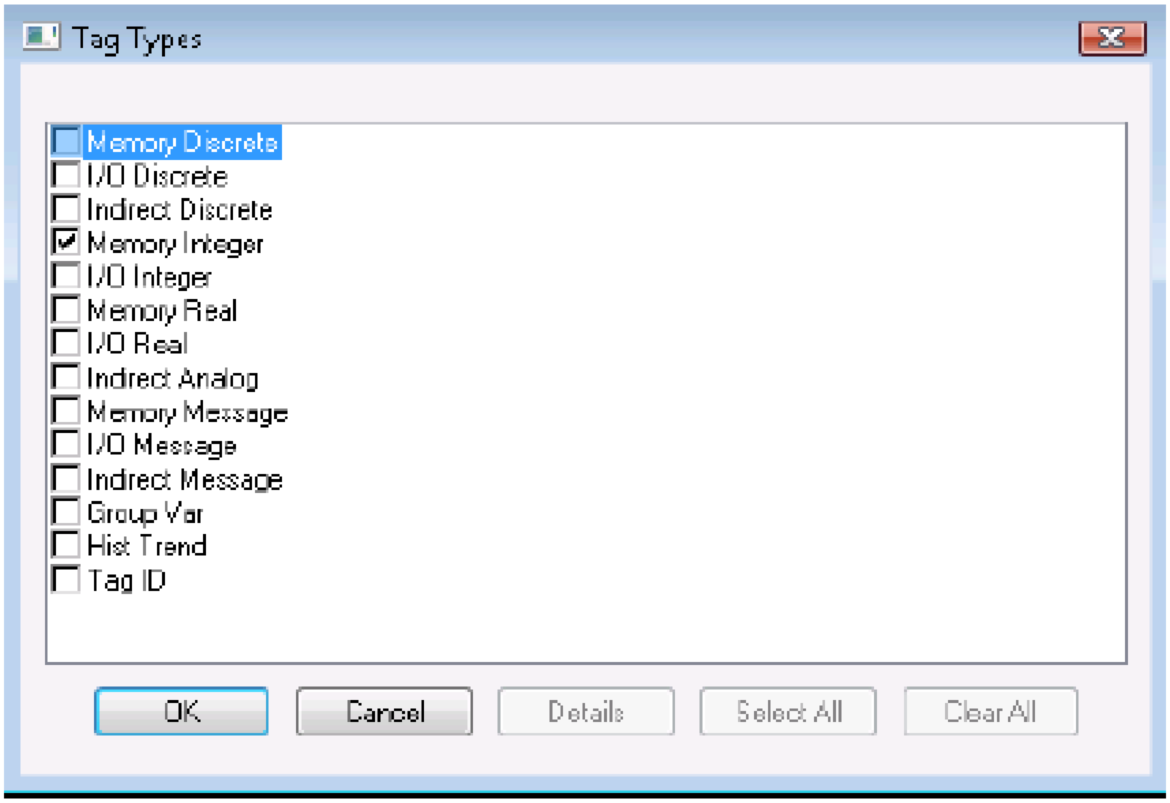
4.5.9 TAGS AND PROPERTIES

A tag represents a data item in an In Touch application. You create tags for those process components whose properties you want to monitor or control with your application. You assign the name and type of tag with the Tag name Dictionary. For some types of tags, you have other options in the Tag name Dictionary to specify additional properties of tags.

When Window Viewer starts an application, it reads the tags from the development repository and places them into run-time memory. The InTouch application communicates with the tags placed into run-time memory using animation links or scripts. The InTouch application tracks the current values and other status information from the component properties assigned to tags.

TAG TYPES

When you define a tag, you assign it to a specific type according to the intended purpose of the tag or the type of data that is associated with it. In the Tagname Dictionary, you use the Tag Types dialog box to assign the tag type to any tag you create.


Tag Types

- [ ] Memory Discrete
- [ ] I/O Discrete
- [ ] Indirect Discrete
- [x] Memory Integer
- [ ] I/O Integer
- [ ] Memory Real
- [ ] I/O Real
- [ ] Indirect Analog
- [ ] Memory Message
- [ ] I/O Message
- [ ] Indirect Message
- [ ] Group Var
- [ ] Hist Trend
- [ ] Tag ID

Fig 4.23 - Tag types

Memory tags define internal system constants and variables within InTouch applications. Memory tags can also act as calculated variables that are accessed by other programs. I/O tags read or write InTouch application data to or from an external source like an I/O Server. External data includes input and output from programmable controllers, process computers, and network nodes.

4.5.10 SCRIPTING AND LOGIC

You can write scripts to monitor and manage aspects of your InTouch applications. A script is a set of programmatic instructions that directs an InTouch application to perform an action. You write InTouch scripts with the Quick Script language. Using Quick Script, you can write scripts that include conditional branching, code looping, and local variables.

Scripts can generally be run in two different ways

- Event-based scripts run once when an event occurs. For example, an event-based script can run after an operator presses a key or a tag value changes.
- Time-based scripts run periodically while a condition is fulfilled. For example, a time-based script can run while a window is open or a button is kept pressed.

A script statement can be a value assignment, a function call, or a control structure. Each statement must end with a semicolon (;).

An In Touch script supports the following types of operators:

Mathematical (addition, subtraction, multiplication)

Boolean (AND, OR, NOT)

Bitwise (bitwise AND, bitwise OR)

Shift (shift left, shift right)

Comparison (less than, equal, greater than)

An InTouch script only supports conditional branching using the IF-THEN-ELSE control structure.

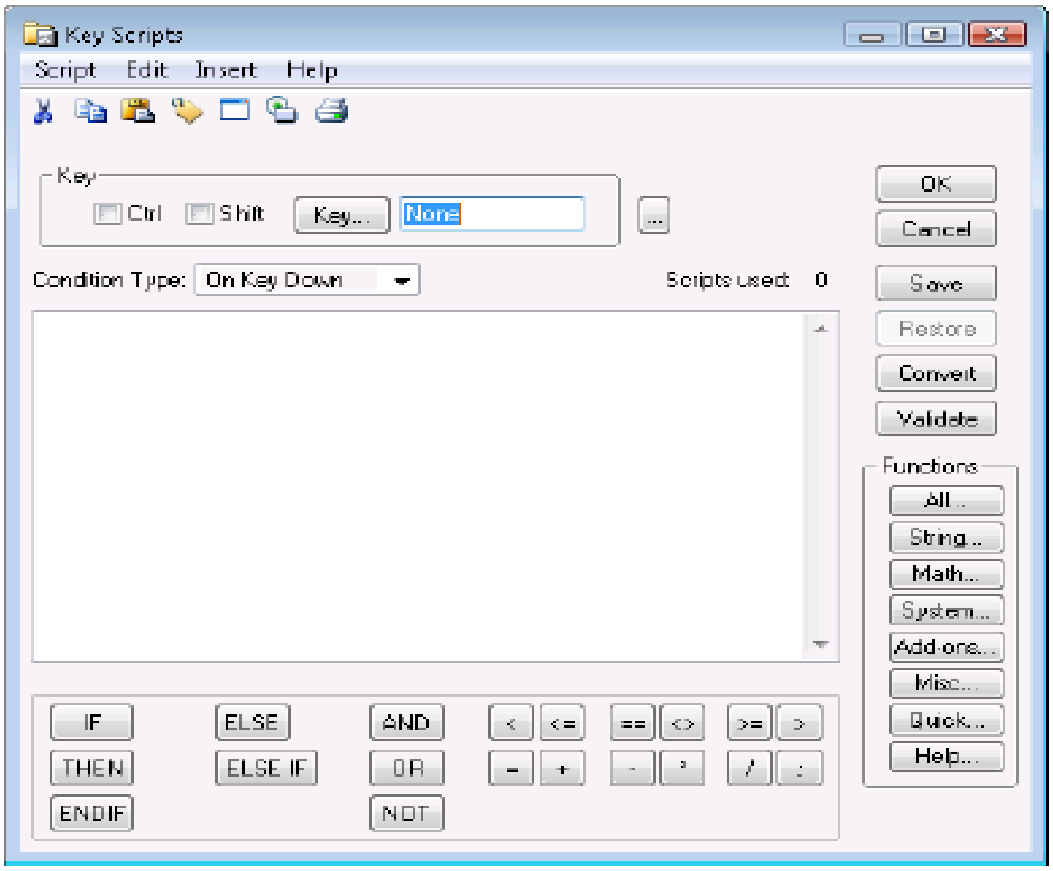


Fig 4.24 - Key scripts

4.5.11 SYMBOL FACTORY

Symbol Factory can be optionally installed with InTouch SCADA . Symbol Factory is a collection of over 4,000 industrial symbols that you can use in your applications. Symbols are organized by their functional categories. After you select a category, you can see the symbols from the category that you can place into an application. Any Symbol Factory symbol can be animated. Symbol Factory provides the most common animation links. If you want to use another type of animation link, you can break the symbol and animate it using standard InTouch animation links. You can add InTouch objects to Symbol Factory. If you add an

InTouch object that has animation links associated with it, the links are also stored with the object. Any tags associated with the object are automatically converted to placeholder tags.

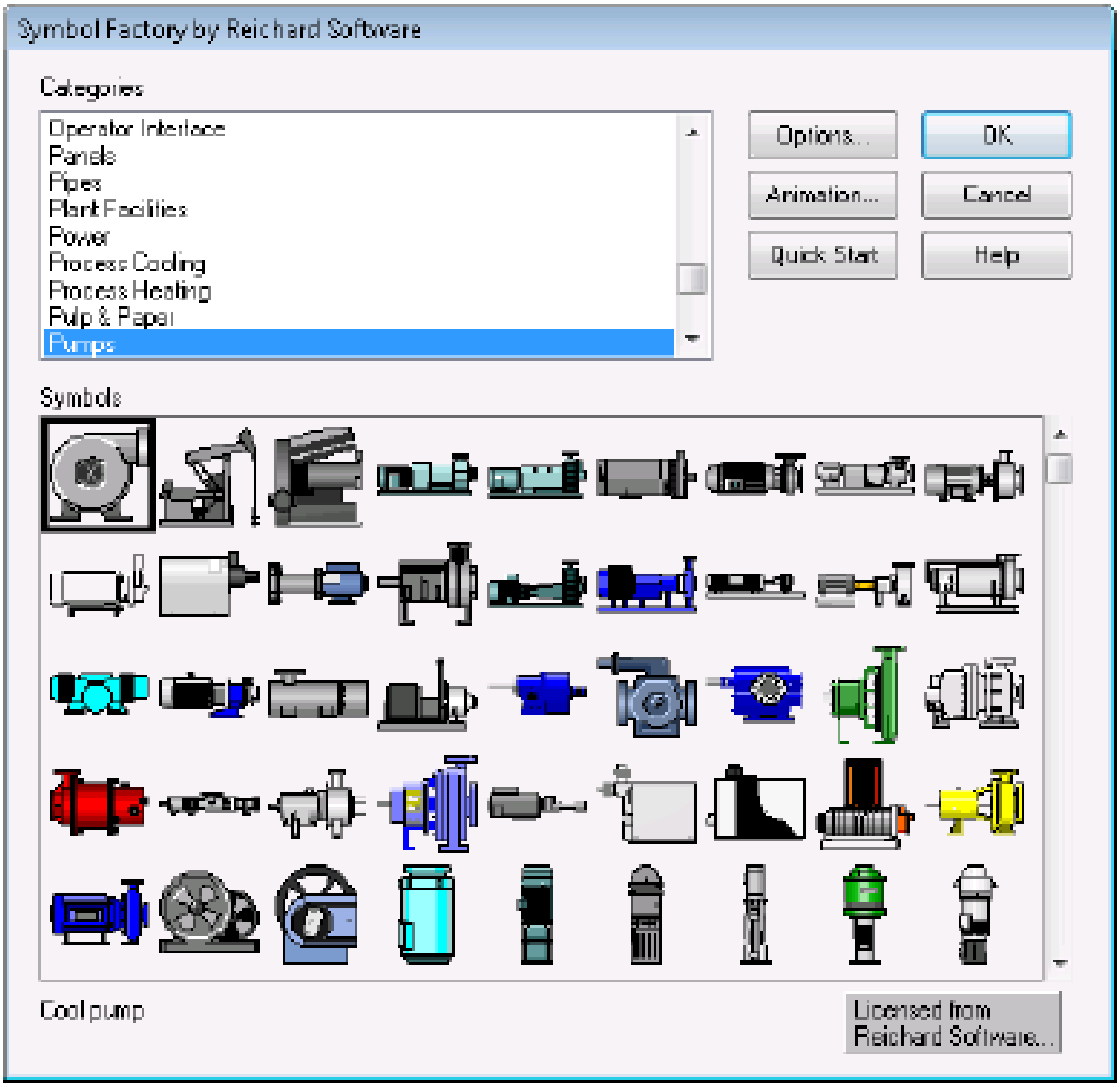


Fig 4.25 - Symbol factory

CHAPTER-5

RESULT AND CONCLUSION

Thus, automating the bearing press machine we can improve the productivity. The assembly time is greatly reduced. For a single job, it takes a total of 36 seconds comprising 30 seconds of machine time and 6 seconds of human time. Many interlocks are provided for fool-proof operation. The infrared sensors are placed near the press to stop the whole process, if there is any interruption for safety purpose. In emergency condition, alarm will be activated and press returns to home position. A separate memory address is given in SCADA for controlling the inputs in case of failure of any field inputs.

Thus this method improves the reliability, accuracy, reduces the maintenance cost and the interlocks provided increases the safety.

APPENDIX-I

PLC OMRON DATASHEET

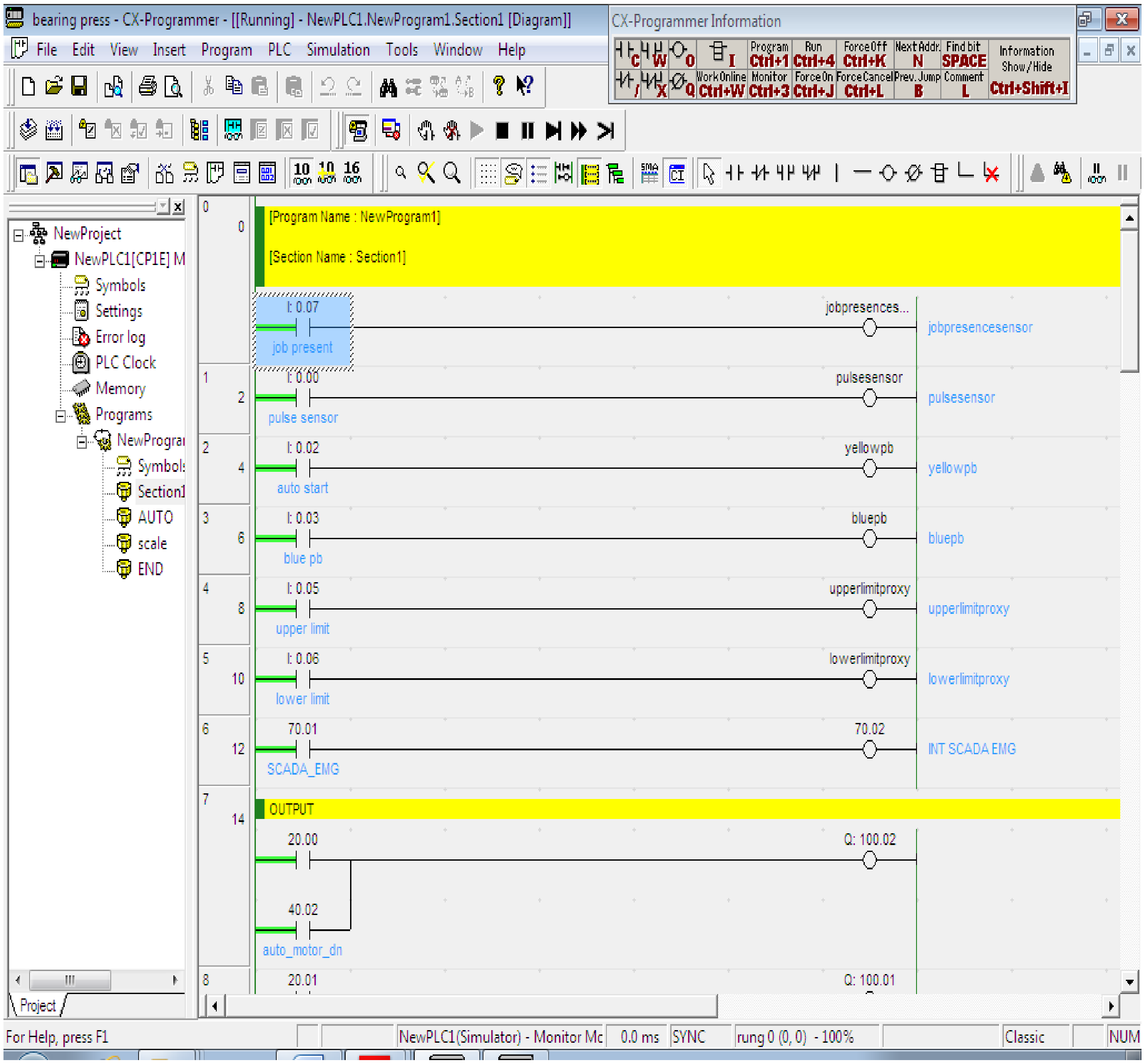
Product name	Specifications						External power supply (24 VDC) (A)	Current consumption (A)		Model	Standards
	Power Supply	Inputs	Outputs	Output type	Program capacity	Data memory capacity		5 V	24 V		
N□□S1-type CPU Units with 30 I/O Points	100 to 240 VAC	18	12	Relay	8K steps	8K words	0.3	0.21	0.07	**CP1E-N30S1DR-A**	CE
	DC24V			Transistor (sinking)			--	0.27	0.02	**CP1E-N30S1DT-D**	
				Transistor (sourcing)			--	0.27	0.02	**CP1E-N30S1DT1-D**	
N□□S1-type CPU Units with 40 I/O Points	100 to 240 VAC	24	16	Relay	8K steps	8K words	0.3	0.21	0.09	**CP1E-N40S1DR-A**	CE
	DC24V			Transistor (sinking)			--	0.31	0.02	**CP1E-N40S1DT-D**	
				Transistor (sourcing)			--	0.31	0.02	**CP1E-N40S1DT1-D**	
N□□S1-type CPU Units with 60 I/O Points	100 to 240 VAC	36	24	Relay	8K steps	8K words	0.3	0.21	0.13	**CP1E-N60S1DR-A**	CE
	DC24V			Transistor (sinking)			--	0.31	0.02	**CP1E-N60S1DT-D**	
				Transistor (sourcing)			--	0.31	0.02	**CP1E-N60S1DT1-D**	

APPENDIX-II

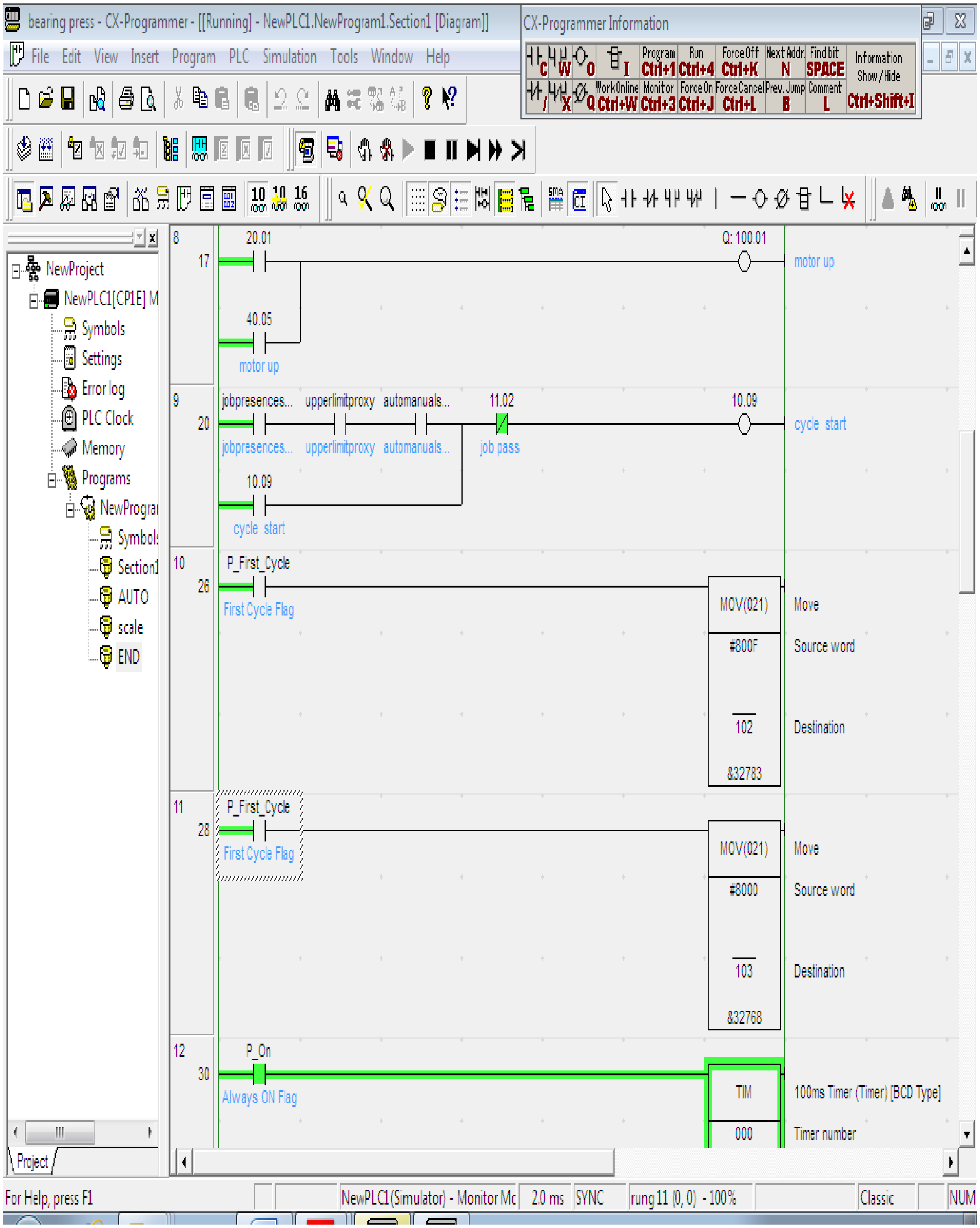
ASSEMBLED HOUSING

APPENDIX-III

PLC LADDER PROGRAM

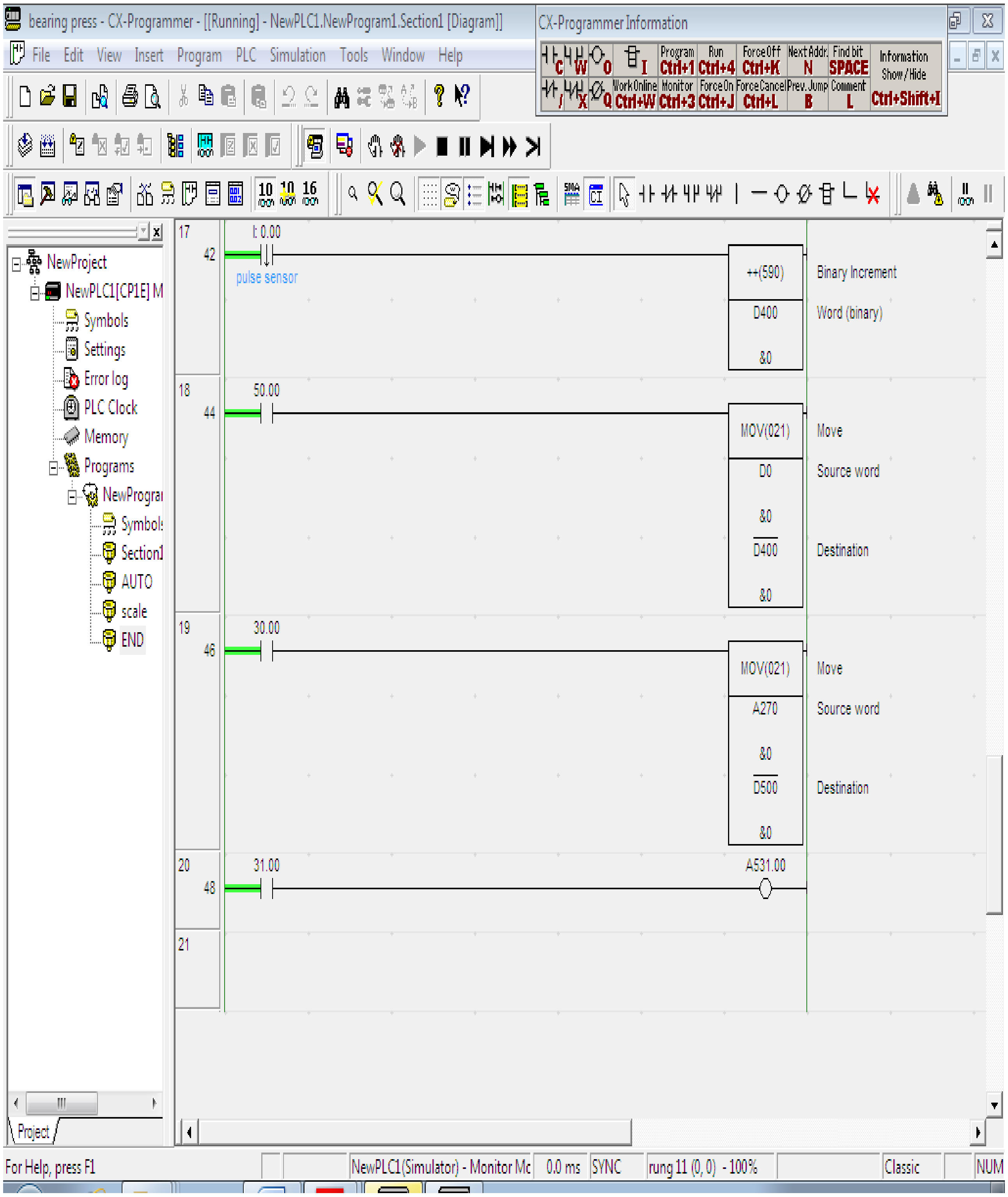

bearing press - CX-Programmer - [[Running] - NewPLC1.NewProgram1.Section1 [Diagram]]
File Edit View Insert Program PLC Simulation Tools Window Help
CX-Programmer Information
NewProject
NewPLC1[CP1E] M
Symbols
Settings
Error log
PLC Clock
Memory
Programs
NewProgra
Symbols
Section1
AUTO
scale
END
Project
For Help, press F1
NewPLC1(Simulator) - Monitor Mc
0.0 ms
SYNC
rung 0 (0, 0) - 100%
Classic
NUM

```
0   0   [Program Name : NewProgram1]
        [Section Name : Section1]
        I: 0.07 job present                 -> jobpresences...  jobpresencesensor
1   2   I: 0.00 pulse sensor                -> pulsesensor      pulsesensor
2   4   I: 0.02 auto start                  -> yellowpb         yellowpb
3   6   I: 0.03 blue pb                     -> bluepb           bluepb
4   8   I: 0.05 upper limit                 -> upperlimitproxy  upperlimitproxy
5   10  I: 0.06 lower limit                 -> lowerlimitproxy  lowerlimitproxy
6   12  70.01 SCADA_EMG                     -> 70.02            INT SCADA EMG
7   14  OUTPUT
        20.00                               -> Q: 100.02
        40.02 auto_motor_dn
8       20.01                               -> Q: 100.01
```


bearing press - CX-Programmer - [[Running] - NewPLC1.NewProgram1.Section1 [Diagram]]

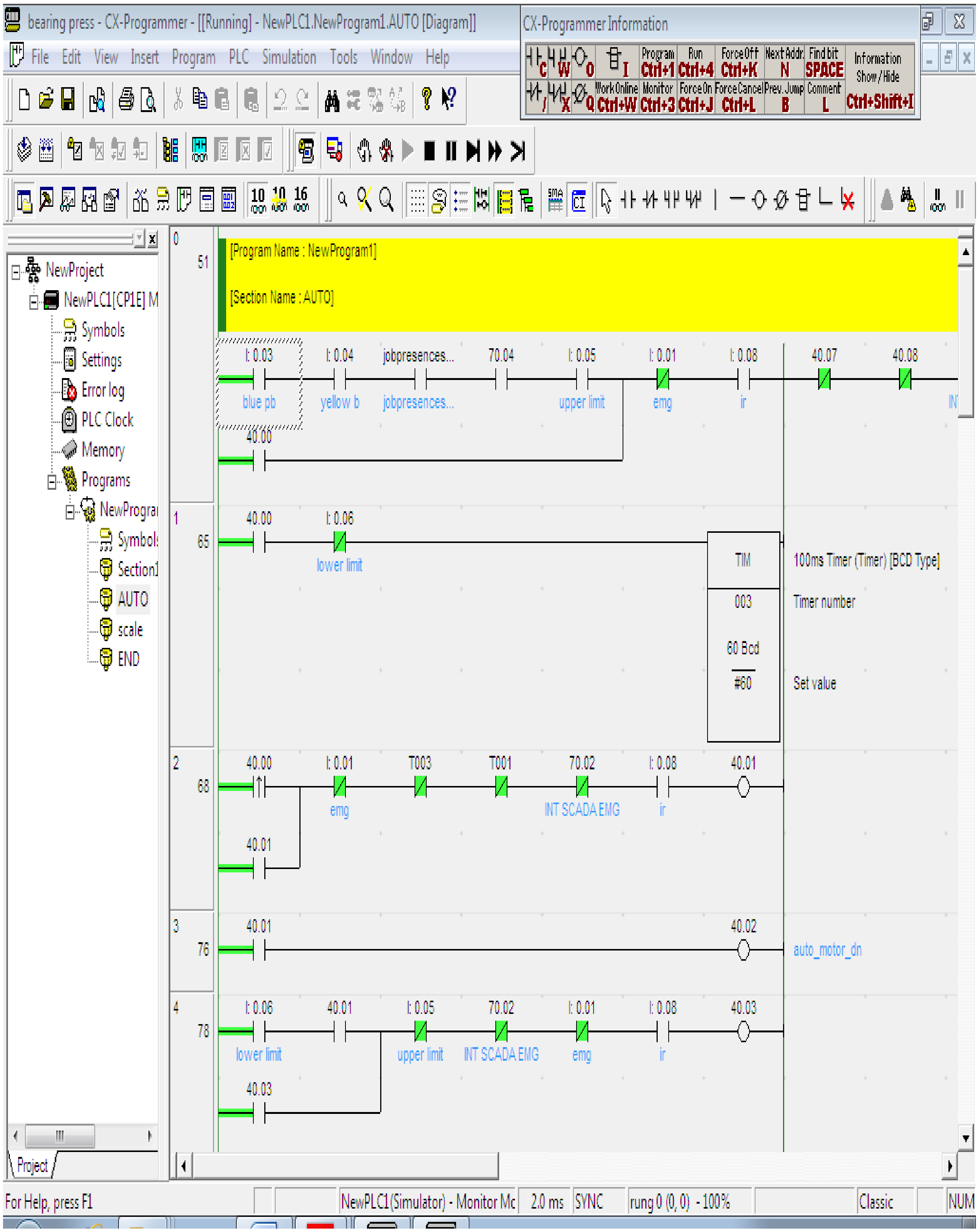
```
8   17   20.01                                        Q: 100.01
         40.05                                                    motor up
         motor up
9   20   jobpresences...  upperlimitproxy  automanuals...  11.02   10.09
         jobpresences...  upperlimitproxy  automanuals...  job pass   cycle start
         10.09
         cycle start
10  26   P_First_Cycle                    MOV(021)   Move
         First Cycle Flag                 #800F      Source word
                                          102        Destination
                                          &32783
11  28   P_First_Cycle                    MOV(021)   Move
         First Cycle Flag                 #8000      Source word
                                          103        Destination
                                          &32768
12  30   P_On                             TIM        100ms Timer (Timer) [BCD Type]
         Always ON Flag                   000        Timer number
```

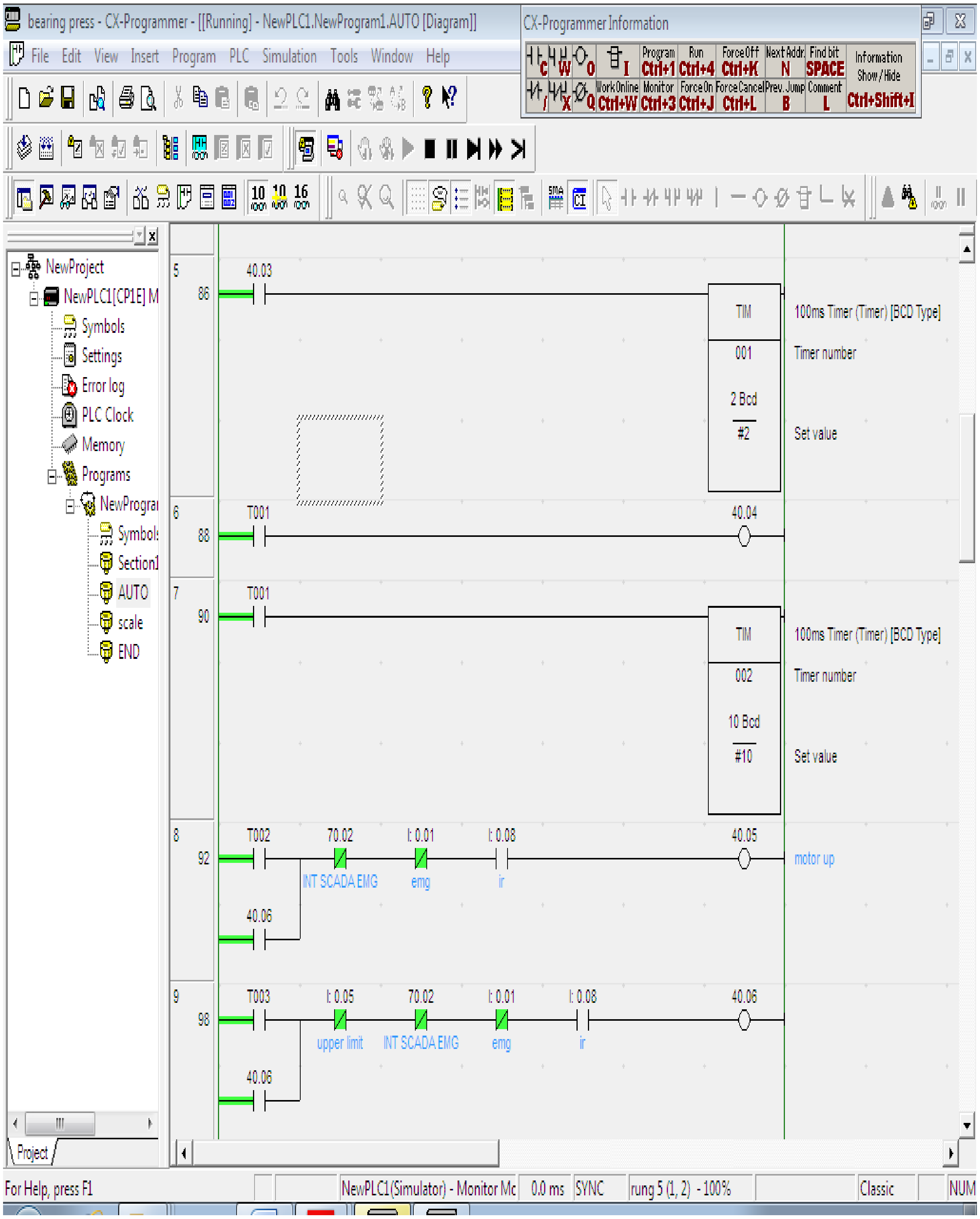
For Help, press F1 | NewPLC1(Simulator) - Monitor Mc | 2.0 ms | SYNC | rung 11 (0, 0) - 100% | Classic | NUM


bearing press - CX-Programmer - [[Running] - NewPLC1.NewProgram1.Section1 [Diagram]]
CX-Programmer Information
File Edit View Insert Program PLC Simulation Tools Window Help
NewProject
NewPLC1[CP1E] M
Symbols
Settings
Error log
PLC Clock
Memory
Programs
NewProgra
Symbols
Section1
AUTO
scale
END
Project

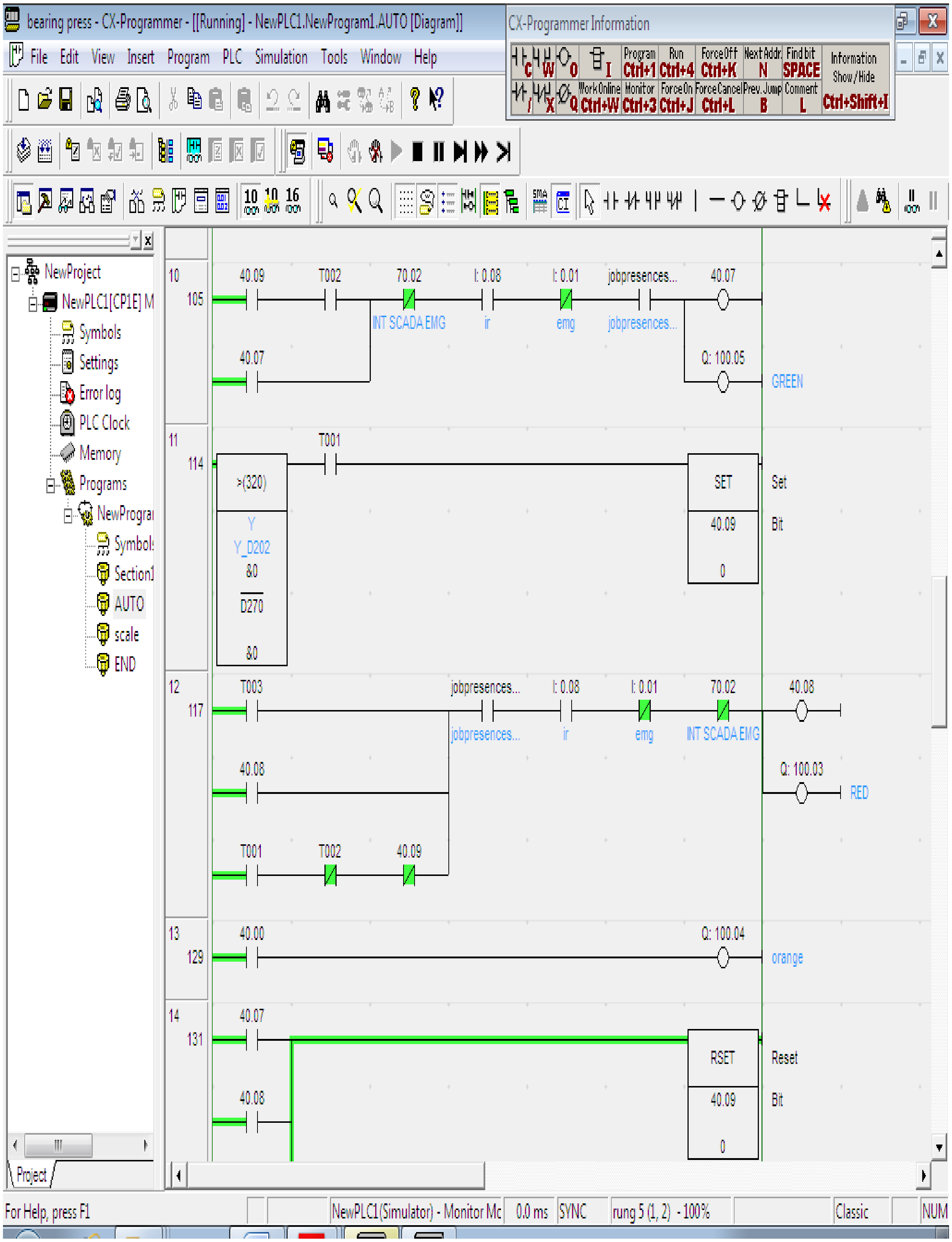
```
17  I: 0.00                                    ++(590)    Binary Increment
42  pulse sensor                               D400       Word (binary)
                                               &0
18  50.00                                      MOV(021)   Move
44                                             D0         Source word
                                               &0
                                               D400       Destination
                                               &0
19  30.00                                      MOV(021)   Move
46                                             A270       Source word
                                               &0
                                               D500       Destination
                                               &0
20  31.00                                      A531.00
48
21
```

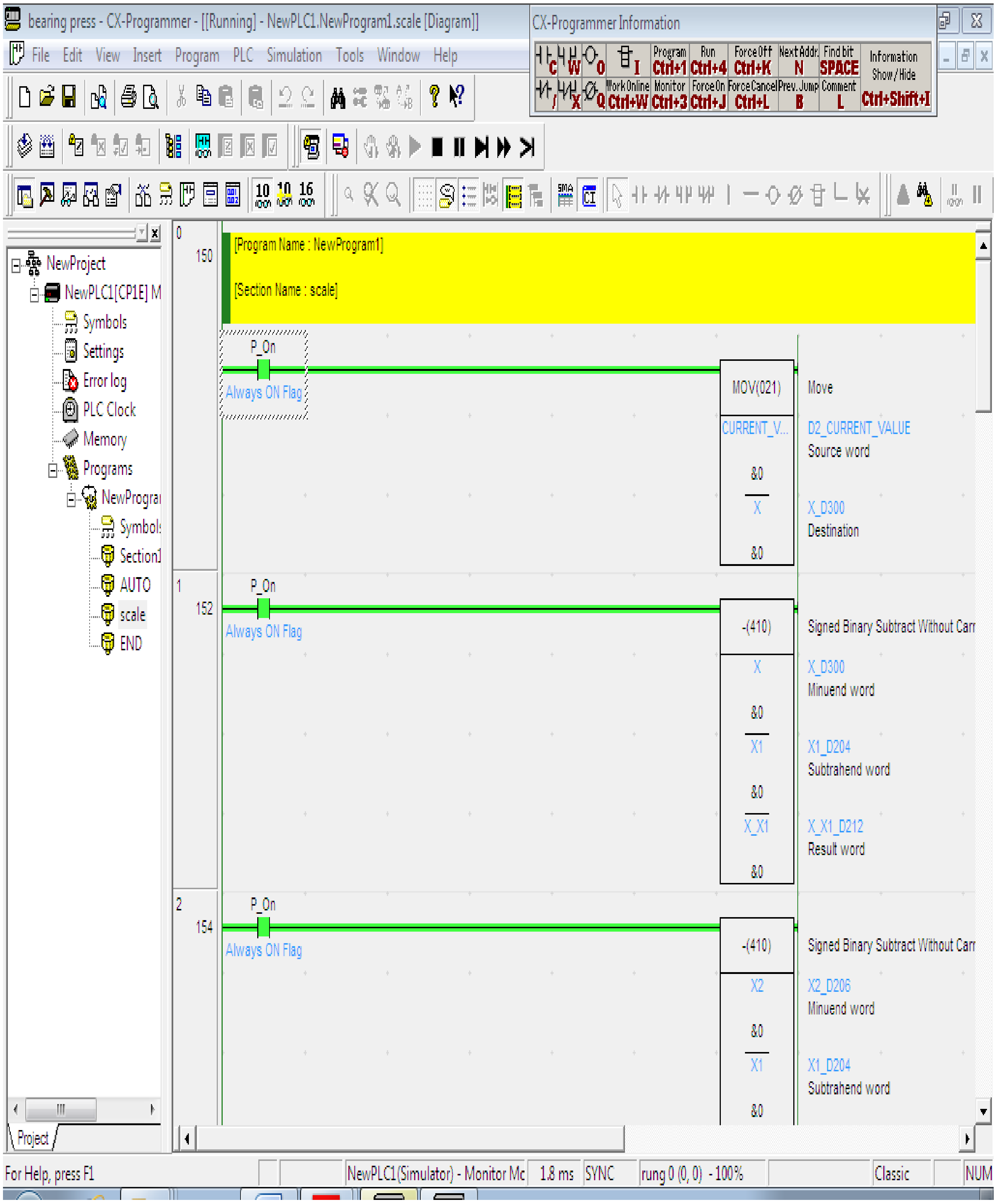
For Help, press F1 | NewPLC1(Simulator) - Monitor Mc | 0.0 ms | SYNC | rung 11 (0, 0) - 100% | Classic | NUM


bearing press - CX-Programmer - [[Running] - NewPLC1.NewProgram1.AUTO [Diagram]]
CX-Programmer Information
File Edit View Insert Program PLC Simulation Tools Window Help
NewProject
NewPLC1[CP1E] M
Symbols
Settings
Error log
PLC Clock
Memory
Programs
NewProgra
Symbol
Section1
AUTO
scale
END
Project
[Program Name : NewProgram1]
[Section Name : AUTO]
I: 0.03 blue pb
I: 0.04 yellow b
jobpresences... jobpresences...
70.04
I: 0.05 upper limit
I: 0.01 emg
I: 0.08 ir
40.07
40.08
40.00
40.00
I: 0.06 lower limit
TIM
003
60 Bcd
#60
100ms Timer (Timer) [BCD Type]
Timer number
Set value
40.00
I: 0.01 emg
T003
T001
70.02 INT SCADA EMG
I: 0.08 ir
40.01
40.01
40.01
40.02 auto_motor_dn
I: 0.06 lower limit
40.01
I: 0.05 upper limit
70.02 INT SCADA EMG
I: 0.01 emg
I: 0.08 ir
40.03
40.03
For Help, press F1
NewPLC1(Simulator) - Monitor Mc
2.0 ms
SYNC
rung 0 (0, 0) - 100%
Classic
NUM


bearing press - CX-Programmer - [[Running] - NewPLC1.NewProgram1.AUTO [Diagram]]
CX-Programmer Information
File Edit View Insert Program PLC Simulation Tools Window Help
NewProject
NewPLC1[CP1E] M
Symbols
Settings
Error log
PLC Clock
Memory
Programs
NewProgra
Symbol
Section1
AUTO
scale
END
Project
For Help, press F1
NewPLC1(Simulator) - Monitor Mc
0.0 ms
SYNC
rung 5 (1, 2) - 100%
Classic
NUM

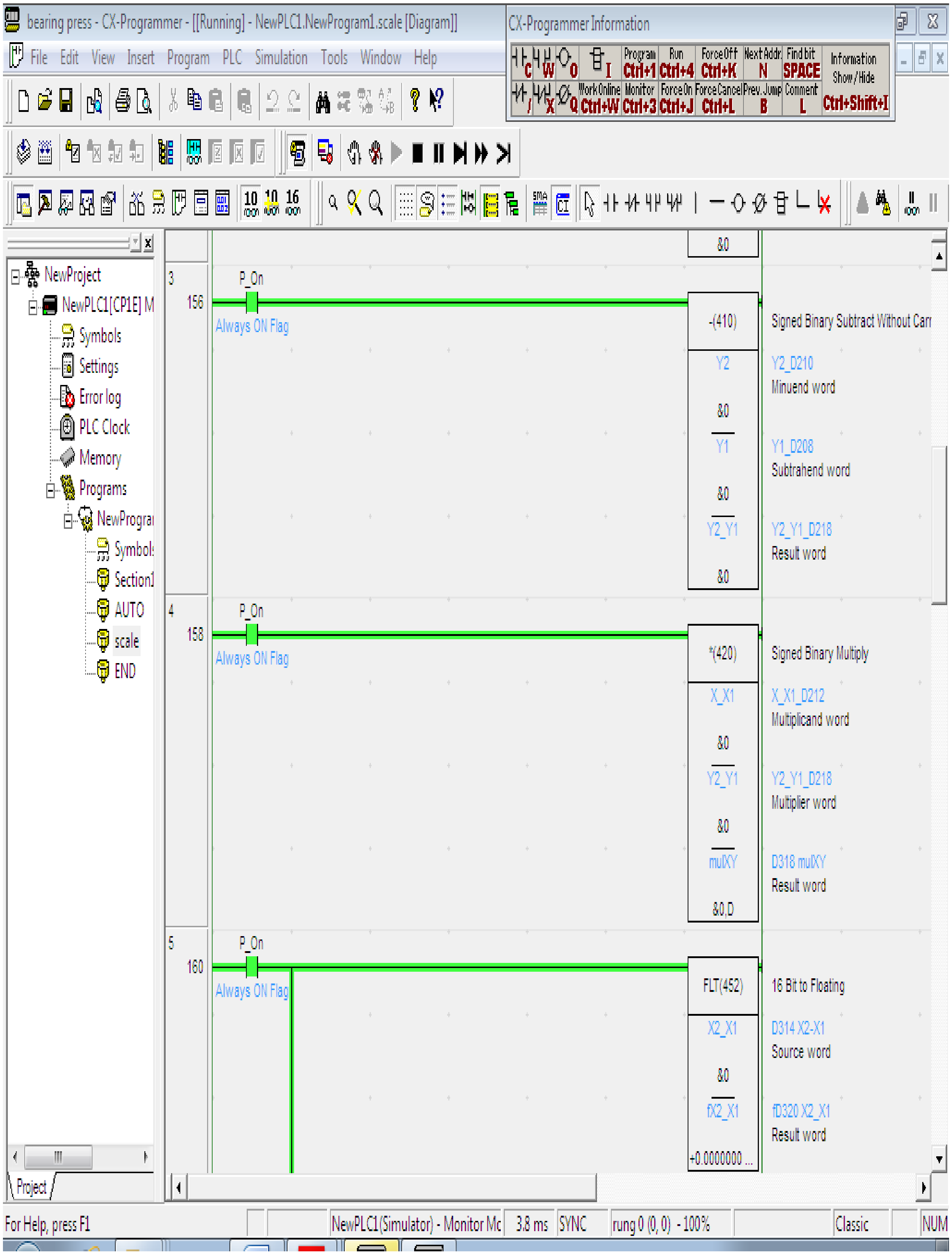
```
5  86   40.03 ---| |------------------------ TIM        100ms Timer (Timer) [BCD Type]
                                             001        Timer number
                                             2 Bcd
                                             #2         Set value
6  88   T001 ----| |----------------------- ( ) 40.04
7  90   T001 ----| |------------------------ TIM        100ms Timer (Timer) [BCD Type]
                                             002        Timer number
                                             10 Bcd
                                             #10        Set value
8  92   T002  70.02          I: 0.01  I: 0.08     40.05
                INT SCADA EMG  emg      ir          motor up
        40.06
9  98   T003  I: 0.05     70.02          I: 0.01  I: 0.08   40.06
                upper limit  INT SCADA EMG  emg      ir
        40.06
```


bearing press - CX-Programmer - [[Running] - NewPLC1.NewProgram1.AUTO [Diagram]]
CX-Programmer Information
File Edit View Insert Program PLC Simulation Tools Window Help
NewProject
NewPLC1[CP1E] M
Symbols
Settings
Error log
PLC Clock
Memory
Programs
NewProgra
Symbol
Section1
AUTO
scale
END
Project
10 105 40.09 T002 70.02 I: 0.08 I: 0.01 jobpresences... 40.07
INT SCADA EMG ir emg jobpresences...
40.07 Q: 100.05 GREEN
11 114 T001 >(320) Y Y_D202 &0 D270 &0 SET 40.09 0 Set Bit
12 117 T003 jobpresences... I: 0.08 I: 0.01 70.02 40.08
jobpresences... ir emg INT SCADA EMG
40.08 Q: 100.03 RED
T001 T002 40.09
13 129 40.00 Q: 100.04 orange
14 131 40.07 RSET 40.09 0 Reset Bit
40.08
For Help, press F1
NewPLC1(Simulator) - Monitor Mc
0.0 ms
SYNC
rung 5 (1, 2) - 100%
Classic
NUM


bearing press - CX-Programmer - [[Running] - NewPLC1.NewProgram1.scale [Diagram]]

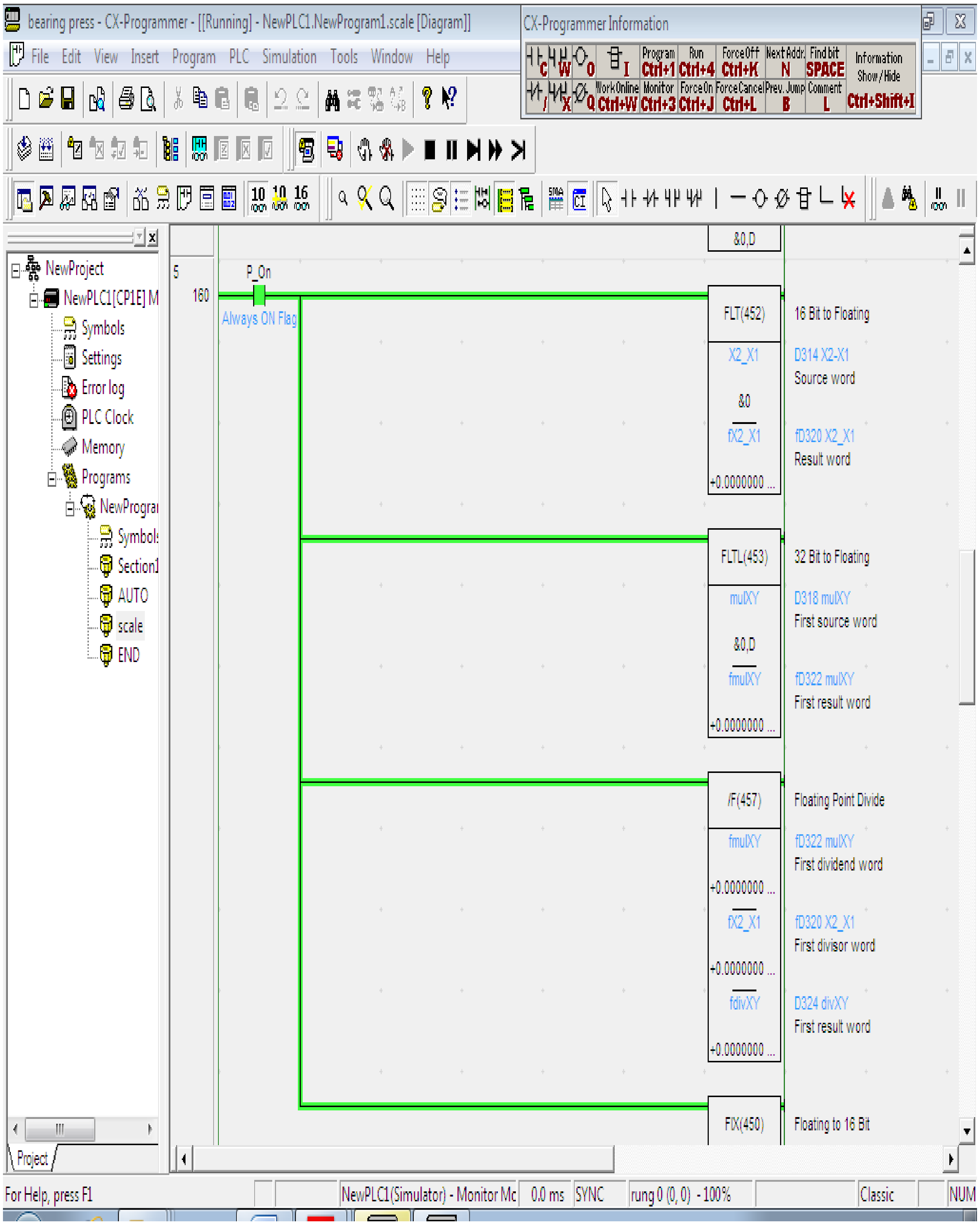
```
0    150   [Program Name : NewProgram1]
           [Section Name : scale]

           P_On                                  MOV(021)      Move
           Always ON Flag                        CURRENT_V...  D2_CURRENT_VALUE
                                                               Source word
                                                 &0
                                                 X             X_D300
                                                               Destination
                                                 &0

1    152   P_On                                  -(410)        Signed Binary Subtract Without Carr
           Always ON Flag                        X             X_D300
                                                               Minuend word
                                                 &0
                                                 X1            X1_D204
                                                               Subtrahend word
                                                 &0
                                                 X_X1          X_X1_D212
                                                               Result word
                                                 &0

2    154   P_On                                  -(410)        Signed Binary Subtract Without Carr
           Always ON Flag                        X2            X2_D206
                                                               Minuend word
                                                 &0
                                                 X1            X1_D204
                                                               Subtrahend word
                                                 &0
```

For Help, press F1 | NewPLC1(Simulator) - Monitor Mc | 1.8 ms | SYNC | rung 0 (0, 0) - 100% | Classic | NUM

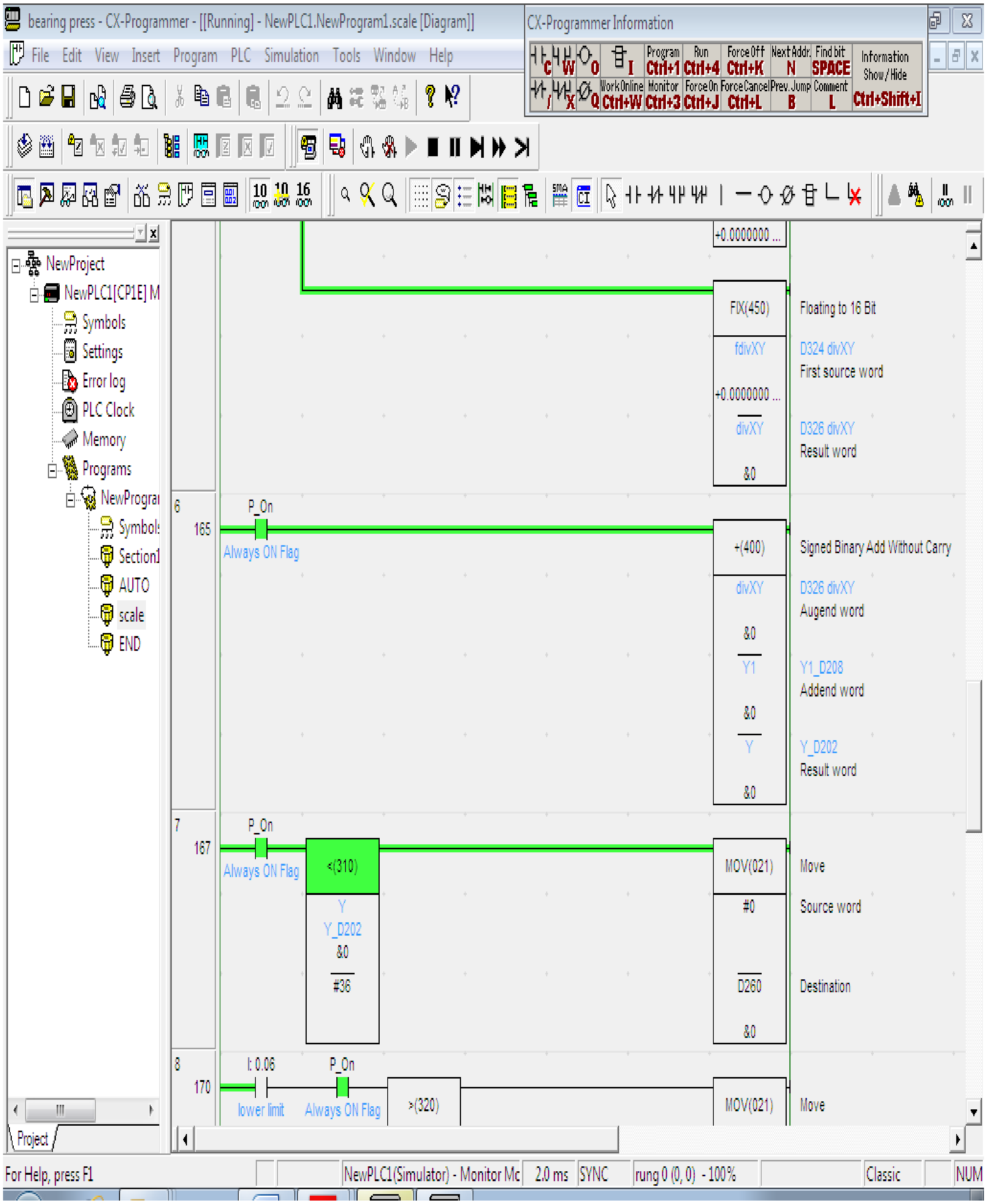

bearing press - CX-Programmer - [[Running] - NewPLC1.NewProgram1.scale [Diagram]]
CX-Programmer Information
File Edit View Insert Program PLC Simulation Tools Window Help
Program Ctrl+1 Run Ctrl+4 ForceOff Ctrl+K Next Addr. N Find bit SPACE Information Show/Hide
WorkOnline Ctrl+W Monitor Ctrl+3 ForceOn Ctrl+J ForceCancel Ctrl+L Prev. Jump B Comment L Ctrl+Shift+I
NewProject
NewPLC1[CP1E] M
Symbols
Settings
Error log
PLC Clock
Memory
Programs
NewProgra
Symbol
Section1
AUTO
scale
END
Project

```
&0

3    P_On                                  -(410)     Signed Binary Subtract Without Carr
156  Always ON Flag                        Y2         Y2_D210
                                           &0         Minuend word
                                           Y1         Y1_D208
                                           &0         Subtrahend word
                                           Y2_Y1      Y2_Y1_D218
                                           &0         Result word

4    P_On                                  *(420)     Signed Binary Multiply
158  Always ON Flag                        X_X1       X_X1_D212
                                           &0         Multiplicand word
                                           Y2_Y1      Y2_Y1_D218
                                           &0         Multiplier word
                                           mulXY      D318 mulXY
                                           &0,D       Result word

5    P_On                                  FLT(452)   16 Bit to Floating
160  Always ON Flag                        X2_X1      D314 X2-X1
                                           &0         Source word
                                           fX2_X1     fD320 X2_X1
                                           +0.0000000 ...  Result word
```

For Help, press F1 | NewPLC1(Simulator) - Monitor Mc | 3.8 ms | SYNC | rung 0 (0, 0) - 100% | Classic | NUM

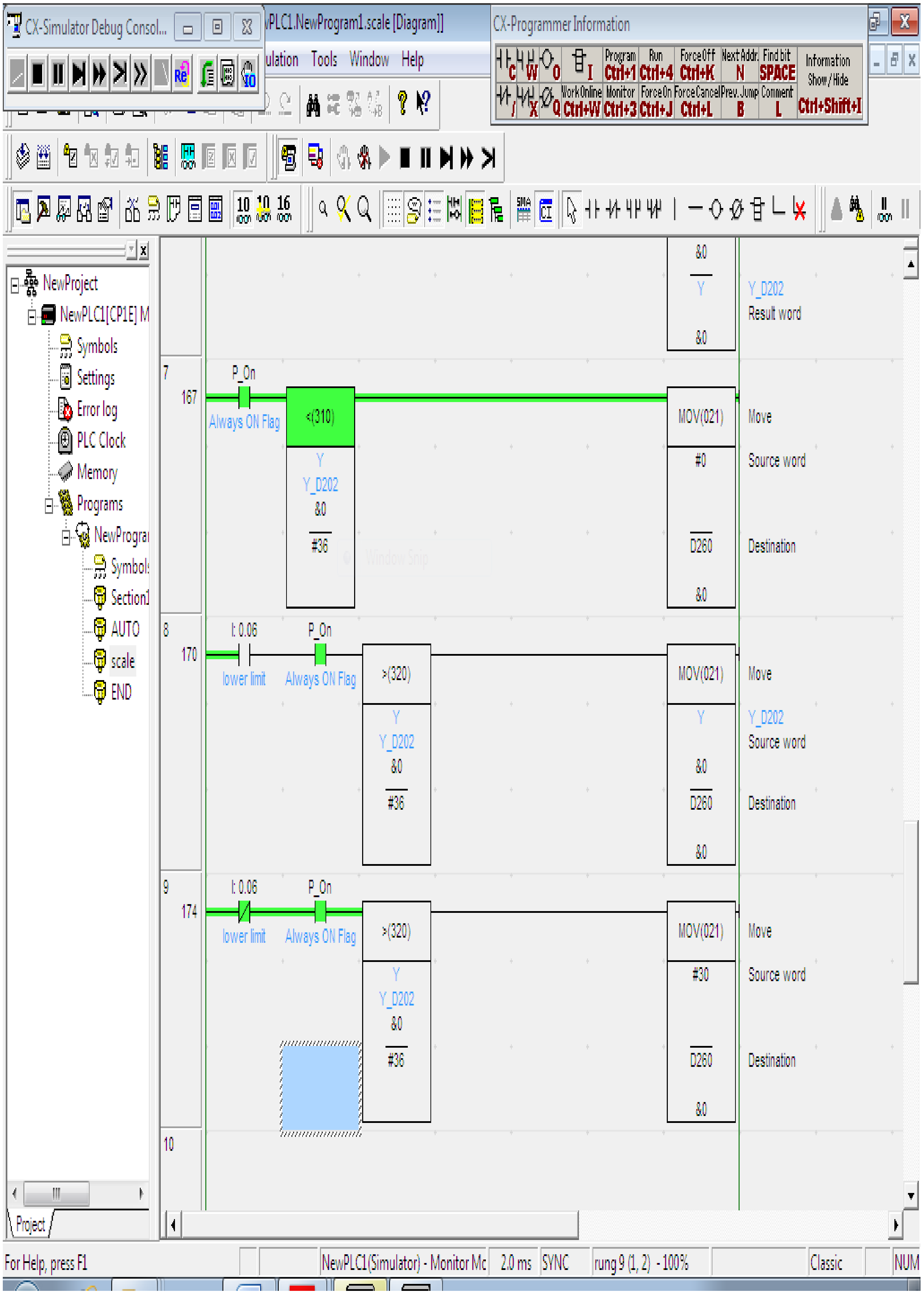

bearing press - CX-Programmer - [[Running] - NewPLC1.NewProgram1.scale [Diagram]]
CX-Programmer Information
File Edit View Insert Program PLC Simulation Tools Window Help
NewProject
NewPLC1[CP1E] M
Symbols
Settings
Error log
PLC Clock
Memory
Programs
NewProgra
Symbols
Section1
AUTO
scale
END
Project
For Help, press F1
NewPLC1(Simulator) - Monitor Mc
0.0 ms
SYNC
rung 0 (0, 0) - 100%
Classic
NUM

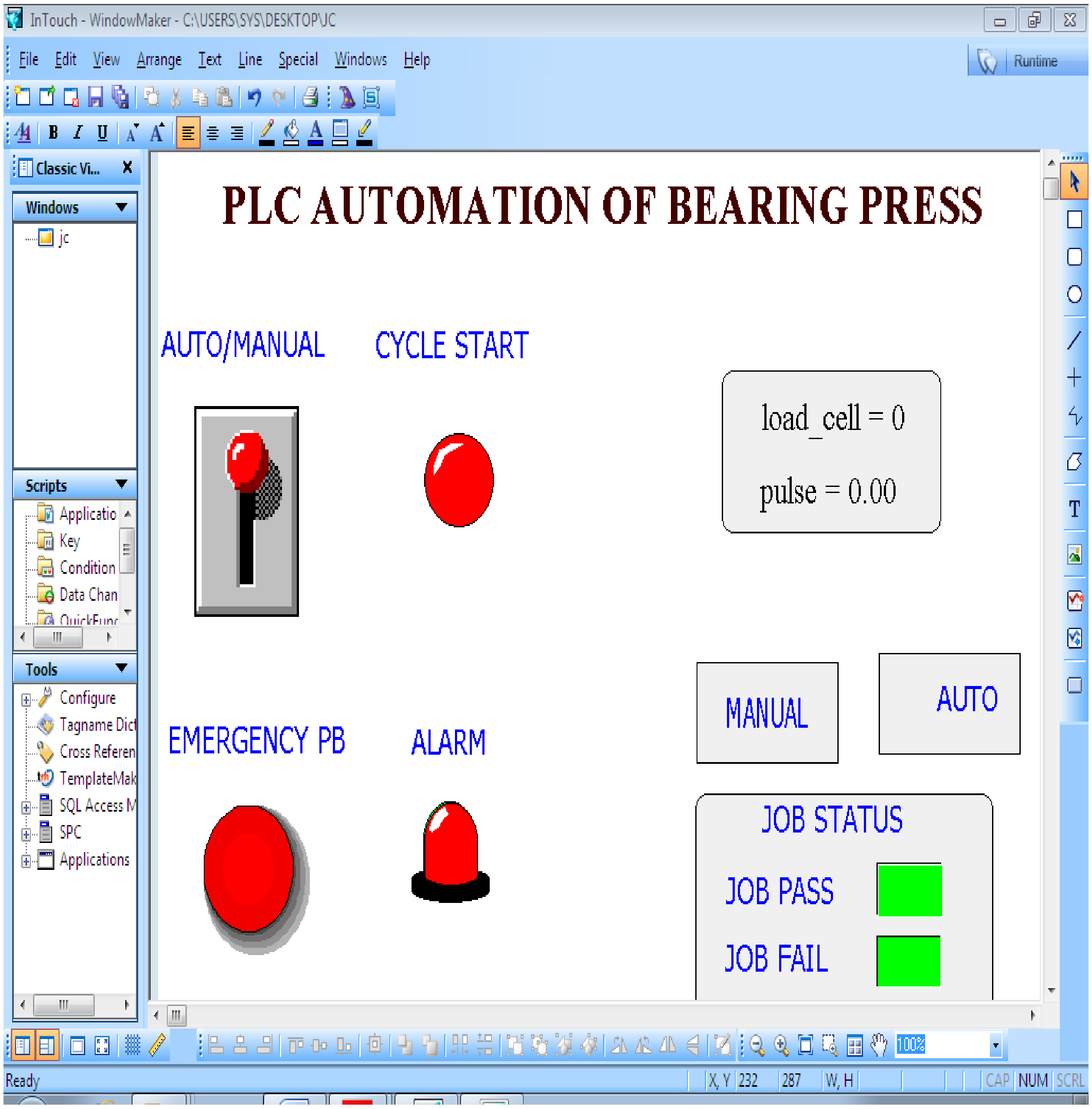
```
&0,D
5    P_On
160  Always ON Flag
FLT(452)      16 Bit to Floating
X2_X1         D314 X2-X1
              Source word
&0
fX2_X1        fD320 X2_X1
              Result word
+0.0000000 ...
FLTL(453)     32 Bit to Floating
mulXY         D318 mulXY
              First source word
&0,D
fmulXY        fD322 mulXY
              First result word
+0.0000000 ...
/F(457)       Floating Point Divide
fmulXY        fD322 mulXY
              First dividend word
+0.0000000 ...
fX2_X1        fD320 X2_X1
              First divisor word
+0.0000000 ...
fdivXY        D324 divXY
              First result word
+0.0000000 ...
FIX(450)      Floating to 16 Bit
```


bearing press - CX-Programmer - [[Running] - NewPLC1.NewProgram1.scale [Diagram]]

```
+0.0000000 ...
FIX(450)      Floating to 16 Bit
fdivXY        D324 divXY
              First source word
+0.0000000 ...
divXY         D326 divXY
              Result word
&0

6  165
P_On
Always ON Flag
+(400)        Signed Binary Add Without Carry
divXY         D326 divXY
              Augend word
&0
Y1            Y1_D208
              Addend word
&0
Y             Y_D202
              Result word
&0

7  167
P_On
Always ON Flag
<(310)
Y
Y_D202
&0
#36
MOV(021)      Move
#0            Source word
D260          Destination
&0

8  170
I: 0.06       P_On
lower limit   Always ON Flag
>(320)
MOV(021)      Move
```

For Help, press F1 | NewPLC1(Simulator) - Monitor Mc | 2.0 ms | SYNC | rung 0 (0, 0) - 100% | Classic | NUM

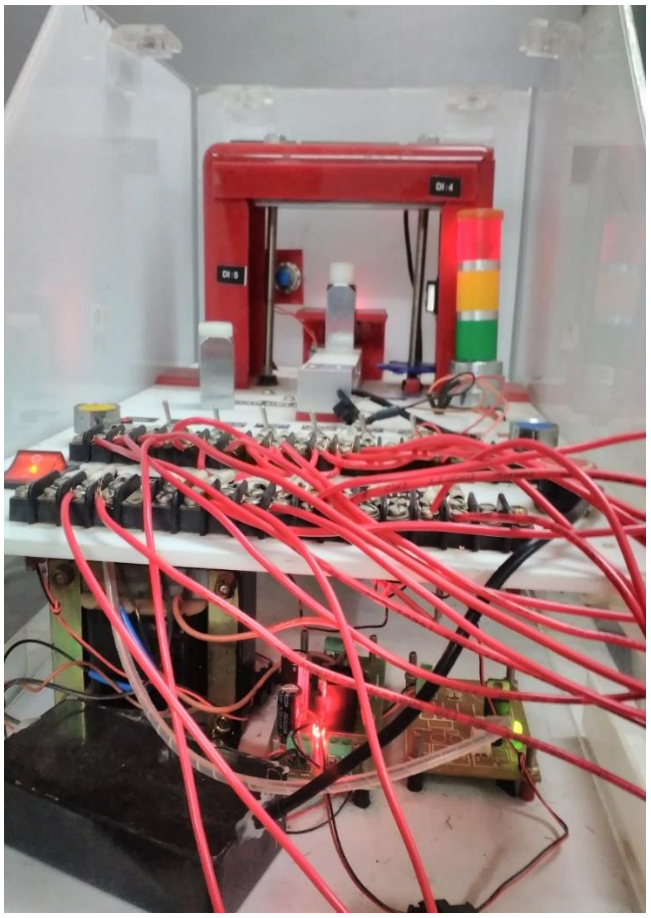

CX-Simulator Debug Consol...
vPLC1.NewProgram1.scale [Diagram]]
CX-Programmer Information
ulation Tools Window Help
Program Ctrl+1
Run Ctrl+4
Force Off Ctrl+K
Next Addr. N
Find bit SPACE
Information Show/Hide Ctrl+Shift+I
Work Online Ctrl+W
Monitor Ctrl+3
Force On Ctrl+J
Force Cancel Ctrl+L
Prev. Jump B
Comment L
NewProject
NewPLC1[CP1E] M
Symbols
Settings
Error log
PLC Clock
Memory
Programs
NewProgra
Symbols
Section1
AUTO
scale
END
Project
&0
Y
Y_D202
Result word
&0
7
167
P_On
Always ON Flag
<(310)
Y
Y_D202
&0
#36
MOV(021)
Move
#0
Source word
D260
Destination
&0
8
170
I: 0.06
lower limit
P_On
Always ON Flag
>(320)
Y
Y_D202
&0
#36
MOV(021)
Move
Y
Y_D202
Source word
&0
D260
Destination
&0
9
174
I: 0.06
lower limit
P_On
Always ON Flag
>(320)
Y
Y_D202
&0
#36
MOV(021)
Move
#30
Source word
D260
Destination
&0
10
For Help, press F1
NewPLC1(Simulator) - Monitor Mc
2.0 ms
SYNC
rung 9 (1, 2) - 100%
Classic
NUM

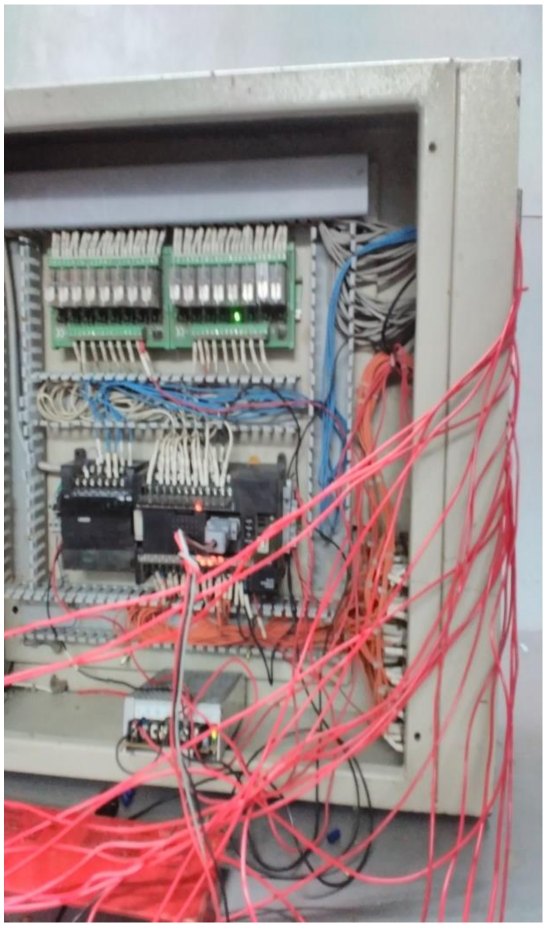


InTouch - WindowMaker - C:\USERS\SYS\DESKTOP\JC
File Edit View Arrange Text Line Special Windows Help
Runtime
Classic Vi...
Windows
jc
Scripts
Applicatio
Key
Condition
Data Chan
QuickFunc
Tools
Configure
Tagname Dict
Cross Referen
TemplateMak
SQL Access M
SPC
Applications
PLC AUTOMATION OF BEARING PRESS
AUTO/MANUAL
CYCLE START
load_cell = 0
pulse = 0.00
MANUAL
AUTO
EMERGENCY PB
ALARM
JOB STATUS
JOB PASS
JOB FAIL
100%
Ready
X, Y 232 287
W, H
CAP NUM SCRL

APPENDIX-IV

PROTOTYPE





REFERENCES

1. M. A. Laughton, D. J. Warne (ed), Electrical Engineer's Reference book, 16th edition, Newnes, 2003 Chapter 16 Programmable Controller
2. Erickson, Kelvin T. (1996). "Programmable logic controllers". Institute of Electrical and Electronics Engineers.
3. Harms, Toni M. & Kinner, Russell H. P.E., Enhancing PLC Performance with Vision Systems. 18th Annual ESD/HMI International Programmable Controllers Conference Proceedings, 1989, p. 387-399.
4. Keller, William L Jr. Grafcet, A Functional Chart for Sequential Processes, 14th Annual International Programmable Controllers Conference Proceedings, 1984, p. 71-96.
5. Gregory K. McMillan, Douglas M. Considine (ed), Process/Industrial Instruments and Controls Handbook Fifth Edition, McGraw-Hill, 1999 ISBN 0-07-012582-1 Section 3 Controllers.
6. Maher, Michael J. Real-Time Control and Communications. 18th Annual ESD/SMI International Programmable Controllers Conference Proceedings, 1989, p. 431-436.
7. W. Bolton, Programmable Logic Controllers, Fifth Edition, Newnes, 2009 ISBN 978-1-85617-751-1.