

Design a Programmable Sequence Controller Utilizing I2C BUS

Raafat. S. Habeeb

Electrical & Computer Eng. Dept., University of Duhok

E-mail: rhabeeb@gmail.com

Mobil: 00964-7705823389

Abstract

New technology makes possible to manufacturing a better type of programmable micro-system. Recently, Chip with the CPU, RAM, ROM, TIMERS, UARTS, and PORTS is available. These types of chip are called microcontroller. Microcontroller/Microprocessor is a programmable device, which executes a loaded program sequentially from top to down (end of program). This type of controller suffer from non-retain last state problem. At any time the electrical power is turned off, the Microcontroller/Microprocessor will start from initial state after power is turned on, losing all intermediate states. If this problem is solved, it might be the Microcontroller/Microprocessor used as programmable sequence controller efficiently. This paper provides a design of sequence controller based on a microcontroller utilizing Serial Electrical Erasable Read Only Memory (SEEPROM) to save the status of process event sequentially. Serial EEPROM is interfaced to the microcontroller via I2C serial BUS, which is a two wire bus using special protocol to transfer data between microcontroller and serial memory. The sequencer is provided with MINI keypad to select the pre-defined time interval through scrolling UP and DOWN the interval values which is monitor in LCD display. This design had been manufactured and tested by controlling the medical syringe process for long time without serious problem.

1.0- Introduction

Single-chip microcontrollers are devices designed for use in products that usually not considered computers, but require the sophisticated and flexible control that a computer can provide. An example of such product is an automatic washing machine. In contrast to microprocessor, microcontrollers are typically integrated RAM, ROM, and I/O, logic circuits designed to execute specific tasks such as Universal Asynchronous Receiver Transmitter (UART), Serial port RS232 and square-wave oscillator (clock), as well as the CPU, onto the same chip. A microcontroller is a computer with most of the necessary support chips onboard.

There are numbers of other common characteristics that define microcontrollers. If a micro system matches a majority of these characteristics, then it can be classified as a 'Microcontroller'. Microcontrollers may be 'Embedded' inside some other device (often a consumer product) so that they can control the features or actions of the product. Another name for a microcontroller is therefore an '*Embedded controller*' dedicated to one task and run one specific program. The program is stored in ROM and generally does not change. A microcontroller may take an input from the device it is controlling and controls the device by sending signals to different components in the device. Most microcontroller circuits are small and low cost compared with microprocessor circuits. The components may be chosen to minimize size and to be as inexpensive as possible. The actual processor used to implement a microcontroller can vary widely. In many products, such as

microwave ovens, the demand on the CPU is fairly low and price is an important consideration. In these cases, manufacturers turn to dedicated microcontroller chips—devices that were originally designed to be low-cost, small, low power, and embedded CPUs. The Motorola 6811 and Intel 8051 are both good examples of such chips [1].

The predominant family of microcontrollers is 8-bit types since this word size has proved popular for the vast majority of tasks the devices have been required to perform. The single byte word is regarded as sufficient for most purposes and has the advantage of easy to interface with the variety of IC memories and logic circuitry currently available. The microcontroller family would have a common instruction subset but family members differ in the amount, and type, of memory, timer facility, port options, etc. possessed, thus producing cost-effective devices suitable for particular manufacturing requirements. Memory expansion is possible with off chip RAM and/or ROM; for some family members there is no on-chip ROM, or the ROM is either electrically programmable ROM (EPROM) or electrically erasable PROM (EEPROM) known as flash EEPROM which allows for the program to be erased and rewritten many times. Additional on-chip facilities could include analogue-to-digital conversion (ADC), digital-to-analogue conversion (DAC) and analogue comparators. Some family members include versions with lower pin count for more basic applications to minimize costs. Since the microcontroller 8051 could not support I2C devices, software has been written to enable the microcontroller to handle this type of communication [2].

1.1- Hardware Overview

The AT89C52 is a low-power, high-performance Complementary Metal Oxide Semiconductor (CMOS) 8-bit microcomputer with 8K bytes of Flash electrical erasable and programmable read only memory (EEPROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 and 80C52 instruction set and pin out. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C52 is a powerful microcomputer, which provides a highly flexible and cost-effective solution to many embedded control applications.

The AT89C52 provides the following standard features: 8Kbytes of Flash, 256 bytes of RAM, 32 I/O lines, three 16-bits timer/counters, a six-vector two-level interrupt architecture, a full-duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89C52 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power-down mode saves the RAM contents but freezes the oscillator, disabling all other chip functions

until the next hardware reset. The AT89C52 has four bi-directional ports designated as P0, P1, P2, and P3. The block diagram of 8052 microcontroller

architecture is shown in Figure 1. The 8051 could not support I2C devices like serial memories EEPROM [3].

1.2 Inter integrated circuit (IIC or I2C) Bus

Commonly referred to as I squared C, the I2C bus or IIC bus was originally developed as a control bus for linking microcontroller and peripheral ICs. The simplicity of a 2-wire bus that combined both address and data bus functions was quickly adopted in many applications such as :

- Telecommunications
- Automotive dashboards
- Energy management systems
- Control and measurement products
- Medical equipment

This method of serial data transmission uses two lines, one for a serial clock (SCL) and the other for serial data (SDA). The SDA line is bi-directional, i.e. data can go up or down it [4].

1.2.1 The Physical I2C Bus

As mentioned earlier, I2C bus is two wires, called SCL and SDA. SCL is the clock line. It is used to synchronize all data transfers over the I2C bus. SDA is the data line. The SCL & SDA lines are connected to all devices on the I2C bus. There needs to be a third wire which is just the ground or 0 volts. There may also be a 5volt wire if power is being distributed to the devices.

SCL and SDA lines are "open drain/collector" drivers. What this means is that the chip can drive its output low, but it cannot drive it high. For the line to be able to go high you must provide pull-up resistors to the 5v supply. There

should be a resistor from the SCL line to the 5v line and another from the SDA line to the 5v line. One set of pull-up resistors for the whole I2C bus is the only that need, not for each device, as illustrated in Figure2 shown below. The value of the resistors is not critical. It is range from 1k8 (1800 ohms) to 4k7 (47000 ohms) used. 1k8, 4k7 and 10k are common values, but anything in this range should work OK. It is recommended 1k8 as this gives you the best performance. If the resistors are missing, the SCL and SDA lines will always be low - nearly 0 volts - and the I2C bus will not work.

The devices on the I2C bus are either masters or slaves. The master is always the device that drives the SCL clock line. The slaves are the devices that respond to the master. A slave cannot initiate a transfer over the I2C bus, only a master can do that. There can be, and usually are, multiple slaves on the I2C bus, however there is normally only one master. It is possible to have multiple masters. Slaves will never initiate a transfer. Both master and slave can transfer data over the I2C bus, but that transfer is always controlled by the master [5].

1.2.2 The I2C Protocol

When the master (your controller) wishes to talk to a slave (our Serial EEPROM in this case). It begins by issuing a start sequence on the I2C bus. A start sequence is one of two special sequences defined for the I2C bus, the other being the stop sequence. The start sequence and stop sequence are special

in that these are the only places where the SDA (data line) is allowed to change while the SCL (clock line) is high. When data is being transferred, SDA must remain stable and not change whilst SCL is high as shown in Figure 3. The start and stop sequences mark the beginning and end of a transaction with the slave device.

Data is transferred in sequences of 8 bits. The bits are placed on the SDA line starting with the MSB (Most Significant Bit). The SCL line is then pulsed high, then low. For every 8 bits transferred, the device receiving the data sends back an acknowledge bit, so there are actually 9 SCL clock pulses to transfer each 8 bit byte of data. If the receiving device sends back a low ACK bit, then it has received the data and is ready to accept another byte. If it sends back a high then it is indicating it cannot accept any further data and the master should terminate the transfer by sending a stop sequence [4].

The standard clock (SCL) speed for I2C up to 100KHz. Philips do define faster speeds: Fast mode, which is up to 400KHz and High Speed mode which is up to 3.4MHz. All of our modules are designed to work at up to 100 KHz. We have tested our modules up to 1MHz but this needs a small delay of a few usec between each byte transferred.

1.2.3 I2C Device Addressing

All I2C addresses are either 7 bits or 10 bits. The use of 10 bit addresses is rare and is not covered here. All of our modules and the common chips will

have 7 bit addresses. This means that up to 128 devices can be connected on the I2C bus, since a 7bit number can be from 0 to 127. When sending out the 7 bit address, it still always needs to send 8 bits. The extra bit is used to inform the slave if the master is writing to or reading from it. If the bit is zero, the master is writing to the slave. If the bit is 1 the master is reading from the slave. The 7 bit address is placed in the upper 7 bits of the byte and the Read/Write (R/W) bit is in the LSB (Least Significant Bit). The placement of the 7 bit address in the upper 7 bits of the byte is a source of confusion for the newcomer. For example, to write to address 21, must actually send out 42 which is 21 moved over by 1 bit left. It is probably easier to think of the I2C bus addresses as 8 bit addresses, with even addresses as write only, and the odd addresses as the read address for the same device. Figure 4a shows the format of device address [5].

1.2.4- Data transfer sequence

A basic Master to slave read or write sequence for I2C follows the following order:

1. Send the START bit (S).
2. Send the slave address (ADDR).
3. Send the Read(R)-1 / Write (W)-0 bit.
4. Wait for/Send an acknowledge bit (A).
5. Send/Receive the data byte (8 bits) (DATA).
6. Expect/Send acknowledge bit (A).
7. Send STOP bit (P).

Note: It could be use 7 bit or 10 bit addresses.

The sequence 5 and 6 can be repeated so that a multi byte block can be read or written.

1.2.5- Data Transfer from master to slave

A master device sends the sequence of signal with START, ADDRESS, and WRITE protocol, then waits for an acknowledge bit (A) from the slave which the slave will only generate if its internal address matches the value sent by the master. If this happens then the master sends DATA and waits for acknowledge (A) from the slave. The master completes the byte transfer by generating a stop bit (P) (or repeated start).

Figure 4b shows the protocol of data transfer from master [4].

1.2.6-Data transfer from slave to master

A similar process happens when a master reads from the slave but in this case, instead of W, R is sent. After the data is transmitted from the slave to the master, the **master** sends the acknowledge signal (A). If instead the master does not want any more data it must send a not-acknowledge which indicates to the slave that it should release the bus. This lets the master send the STOP or repeated START signal. Figure4c shows the protocol of data transfer to master [4].

2. Materials and Methods

The serial EEPROM supports a bidirectional two wire bus and data transmission protocol. A device that send data onto the bus is defined as transmitter, and a device receiving data as receiver. The bus has to be controlled by a master device which generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions, while the serial EEPROM works as slave. Both master

and slave can operate as transmitter or receiver but the master device determines which mode is activated, up to eight 1Kb/2Kb serial EEPROM can be connected to the bus, selected by the A0, A1 and A2 chip address inputs.

The interfacing of 2K byte serial EEPROM with microcontroller is shown in Figure 5. The memory chip address inputs A0, A1 and A2 of serial EEPROM must be externally connected to either VCC or ground (VSS), assigning to each 24C01A/02A/04A a unique programmable address. Up to eight 24C01A or 24C02A devices and up to four 24C04A devices may be connected to the bus. Chip selection is then accomplished through software by setting the bits A0, A1 and A2 of the programmable slave address to the corresponding hard-wired logic levels of the selected 24C01A/02A/04A. After generating a START condition, the bus master transmits the slave fixed address consisting of a 4-bit device code (1010) for the 24C01A/02A/04A, followed by the programmable chip address bits A0, A1 and A2. SDA and SCL lines are connected to port0.0 and port0.1 via a pull up resistor 5K Ω [6].

2.1-Controller Design

The design of programmable system normally is consisting of two parts. The first part is concerning with the hardware design, while the second part is concerning with the software design. The complexity and cost of each part are the factors that the designer will decide in which way where have to emphasis. In this project it is emphasized on software since the 8051 microcontroller is not I2C support [7].

2.1.1 Hardware Design

Many of the applications of microcontroller fall into two categories: Open-Loop or Closed-Loop control systems. Open loop, often called sequential control, is used in applications where the process or device being controlled is characterized by a sequence of state. That is, the application is *event-driven*. An example is a automatic washing machine or vending machine that accept various value coins, recognizes product, selection, vends the product, finds the price, and returns the correct change.

Closed-loop control is characterized by the use of real-time monitoring of process to achieve effectively continuous control. The output of the process is monitored using various transducers and A/D converters and the process is modified continuously [7].

In this application, an event sequential controller is designed; ten sequential external events are controlled according to verification of 8 conditions as shown in Figure 6 below. Port 0 is configured as output port and is used to drive 8 different processes; port 2 is used to display the status of above process. Port 1 and p3.0 – p3.2 and p3.7 are configured as input port that is used to monitor the process conditions. Logic 0 is considered as active logic of port 0 to avoid the glitches (jitter) on this port during RESET or after turn the controller ON. Octal inverter (open collector type) is used to drive a relay bank driver. Serial EPROM of 2 Kb is used to save the last state which has been served before the Electrical power is off. This memory chip is interfaced to microcontroller through ports P2.0 and P2.1 via pull up resistors, Serial Data (SDA), which is a bidirectional signal,

and Clock (SCL) signal are connected to P2.1 and P2.0 respectively [8].

2.1.2 - Software Design

The function of main program is to read the content of serial EEPROM to find the status of the controller was before power shutdown or system reset by reading state number, then the program execute the pre-fetching state. Before transferring to execute next state, the program updates the serial EEPROM by writing the executing state number. When the jumper switch throws to maintenance position, then the program check the all-state sequentially from state 1 up to state 10 each time pressing the push button at P3.6 pin [9].

Event sequential process controller is a process variable dependent. This type of process is performing the specific function continuously until the process variable is true, at this case the controller enforce the process to jump to and execute a next state and so on up to the end of all process state. The second type is time sequential process, which is time depending process. This means that the execution of process is depending on a pre-defined time period until time reach zero. At this point, the controller enforces the process to jump and execute next state and so on up to the end of all state. The first type of sequential controller is taken into consideration during this research. The software have not been used the interrupt feature of microcontroller, so it is not necessary to enable the interrupt system of microcontroller [10]. The software is designed in module structure, which is a main program and many subroutines are invoked by it. The flow chart of main program is shown in Figure 7.

The following is the subroutine that are invoked by main program, all subroutines have a common feature that are returned a carry flag CY which indicate the statues of writing into or readings from serial memory SEEPROM.

READ STAGE NO:

Read from a specific serial EEPROM location with stage number. Return CY=1 to indicate write time over. **REDBYT** subroutine is invoked by this routine. This routine is repeated 5 times to insure correct reading of serial EEPROM Registers A and B are destroyed. The flow chart is shown in Figure 8

REDBYT:

Serial EEPROM Random Read Function, called with programmable address in A. Byte address in R2 return data in A. Return CY, if CY=1, it indicates that the bus is not available or that the addressed device failed to acknowledge. Three subroutines, **SHOUT**, **REDCRNT**, and **START**, are invoked by this routine. Registers A,B and R2 are used. The flow chart is shown in Figure 8.

WRITE STAGE NO:

Write into a specific serial EEPROM location with stage number. Return CY=1 to indicate write time over. **WRITBYT** subroutine is invoked by this routine. This routine is repeated 5 times to insure correct writing into serial EEPROM. Registers A and B are destroyed. The flow chart is shown in Figure 9

WRITBYT:

Serial EEPROM Byte Write function. Call with programmable address in A, Data in register R1. Return CY =1, it

indicates that the bus is not available or the addressed device failed to acknowledge. Two subroutines, **START** and **SHOUT**, are invoked by this routine. A is destroyed. The flow chart is shown in Figure 9.

START: This routine is recalled in REDBYT and WRITBYT routines. It is sending a START signal, define as high-to-low SDA with SCL high. Return with SCL, SDA low. CY=1, the bus is not available. None of registers are used The flow chart is shown in Figure 11.

SHOUT: This routine is recalled in REDBYT and WRITBYT routines. Its function is to shift out a byte to the serial EEPROM, most significant bit **MSB** first. SCL and SDA expected low on entry. Return with SCL low, A is holding data. A reg. will destroy. The flow chart is shown in Figure 10

REDCRNT: This routine is recalled in REDBYT routine. The function of this subroutine is to read programmable address, call with programmable address in A. Return data in A and CY. If CY=1, then it indicates that the bus is not available or the addressed device failed to acknowledge. **SHIN** subroutine is invoked. The flow chart is shown in Figure 10

SHIN: This subroutine is recalled in REDCRNT routine. The function of this subroutine is to shift in a byte from the SEEPR0M, most significant bit **MSB** first. SCL expected low on entry. Return with SCL low. Returns received data byte in A. The flow chart is shown in Figure 11.

3. Discussion and Conclusion

A microcontroller with serial EEPROM and suitable program had been constructed as PCB card as shown in photograph (Figure12) and put into real test. A medical syringe manufacturing plant had been controlled by this controller. It was running for more than 6 months without any malfunction or snakes.

A microcontroller is a specialized form of microprocessor that is designed to be self-sufficient and cost-effective, where a microprocessor is typically designed to be general purpose (the kind used in a PC). Microcontrollers are frequently found in automobiles, office machines, toys, and appliances. Also, a microcontroller is part of an embedded system, which is essentially the whole circuit board. The difference is that microcontroller incorporates features of microprocessor (CPU, ALU, Registers) along with the presence of added features like presence of RAM, ROM, I/O ports, counter etc. Here microcontroller control the operation of machine using fixed program stored in ROM that doesn't change with lifetime. The advantages of microcontroller over microprocessor are low cost to manufacture, easy to implement, and fast compare with microprocessor. Atmel 89S52 microcontroller has a substantial advantage over Atmel 89C52 in term of programming. 89S52 chip has in circuit programming features, so it is not need to an external programmer. Unfortunately, this chip is not available in our local market, so 89C52 microcontroller has been used as a fait accomplice.

A programmable sequential controller card as shown in Figure10 based on 89c52 microcontroller was built and tested. This card was substituted a big size control board which contain a huge number of relays and rotary sequencer. This controller had been put into real operation in sterilization of medical injection process for more than year without any serious problem, only one malfunction of the controller happened during this period due to spike happened in electrical power supply network. This causes to damage the serial EEPROM. This controller could be used in many home appliances such as washing machine both for dishes or clothing. Cooling/heating system and etc.

8052 Microcontroller Block Diagram

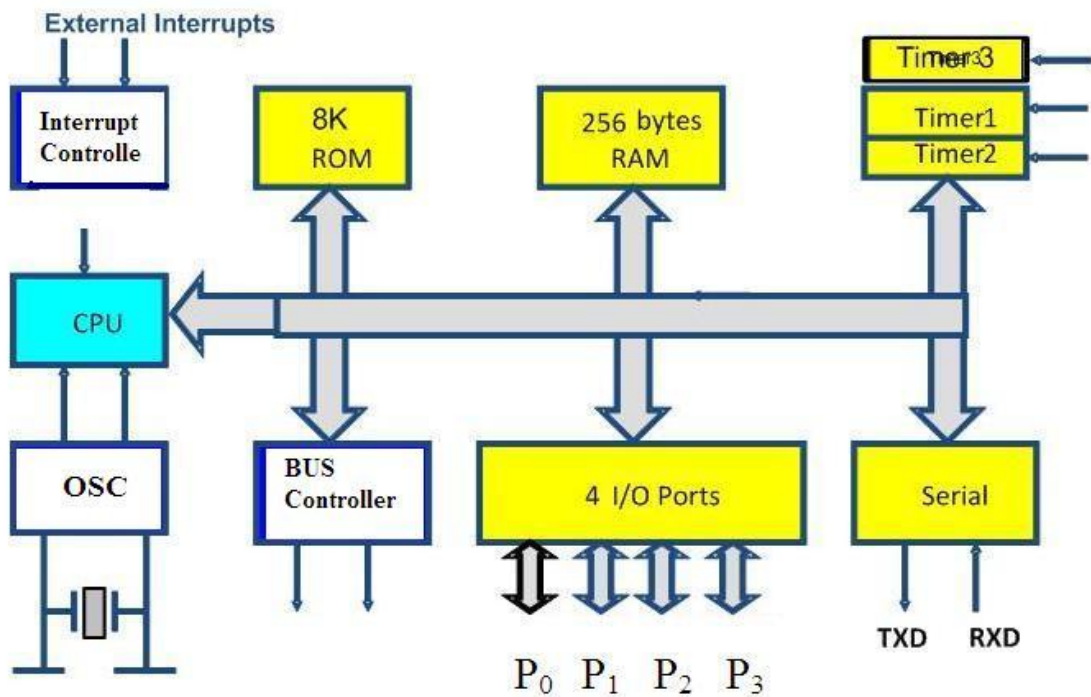


Figure 1 Block diagram of 8052 Hardware Architecture

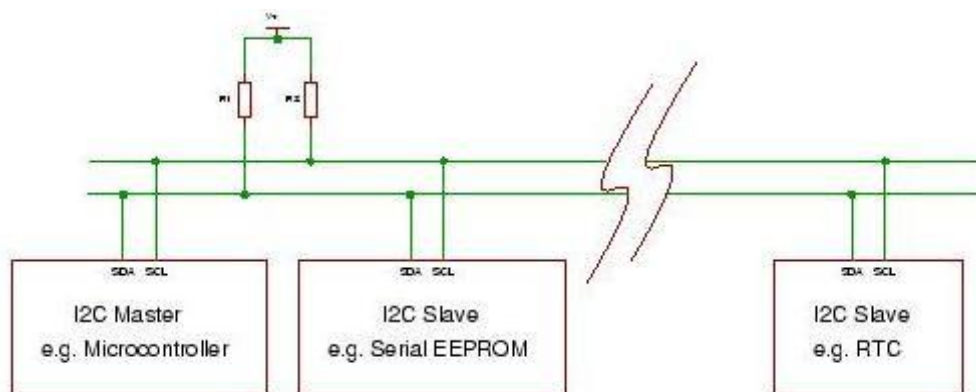


Figure 2 Interfacing Serial EEPROM with I2C

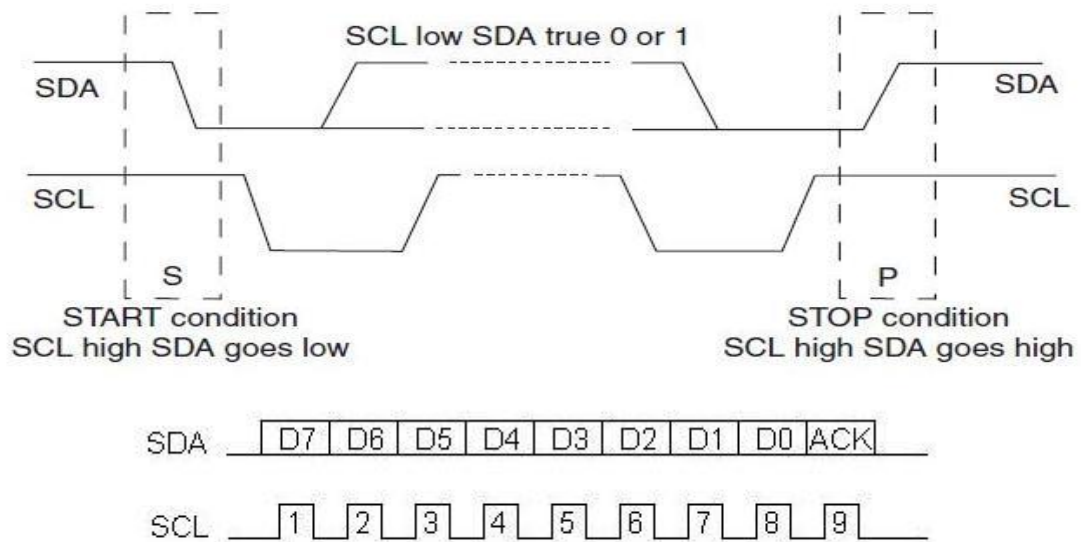
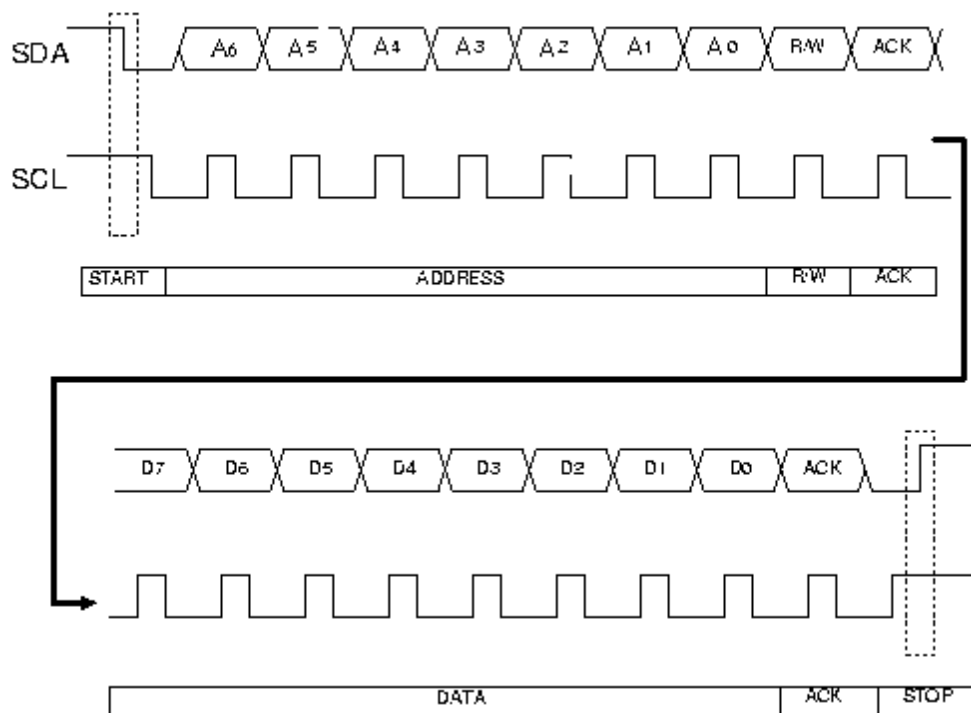




Figure 3 Timing diagram of I2C



(a) Typical SDA and SCL Signals and address format



 sent by master

 sent by slave

(b) Data transfer from master protocol



 sent by master

 sent by slave

(c) Data transfer to master protocol

Figure 4

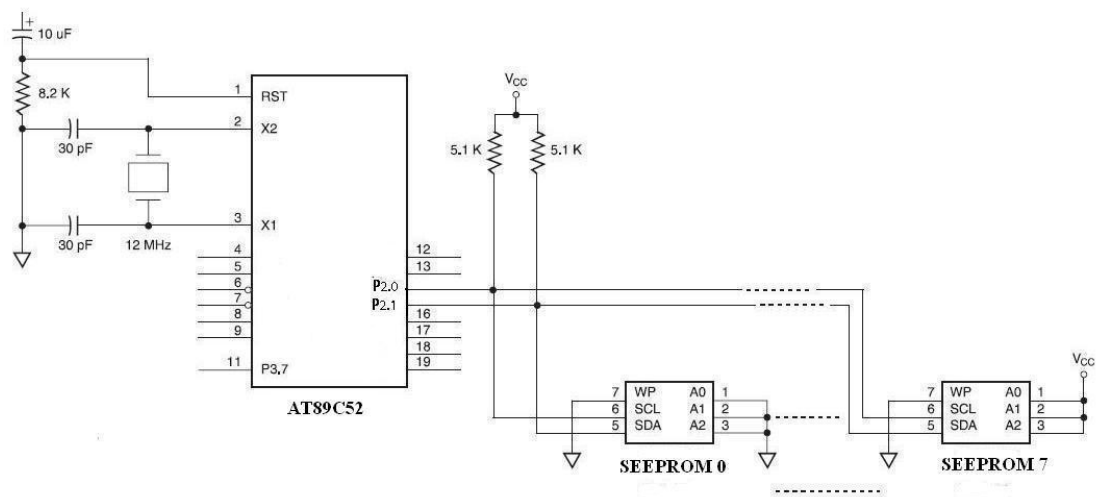


Figure 5 Multi Serial EEPROM(SEEPROM) interfacing

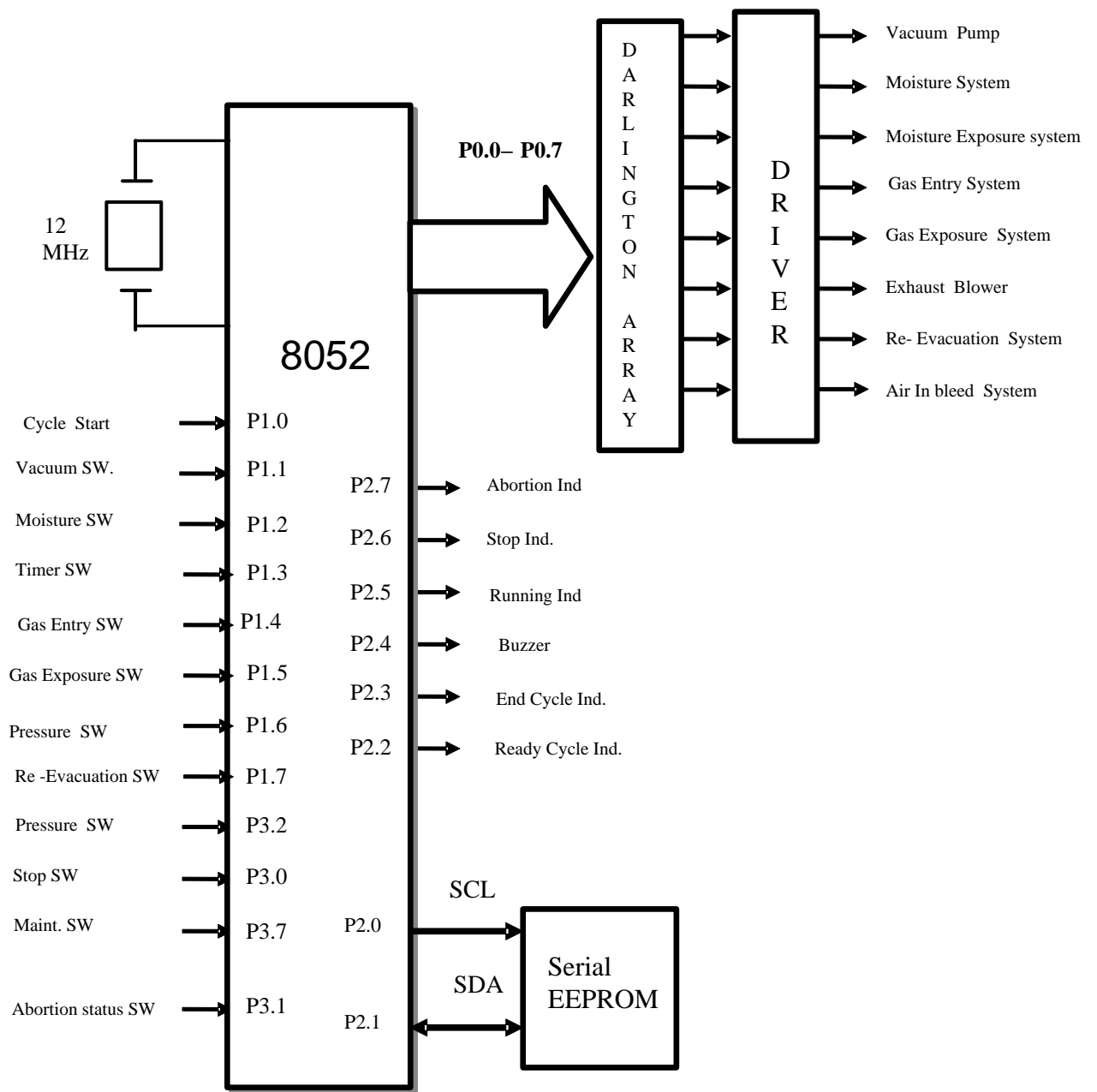


Figure 6. Event sequential controller

Legend :

M: Manual

A: Auto

Y: YES

N: NO

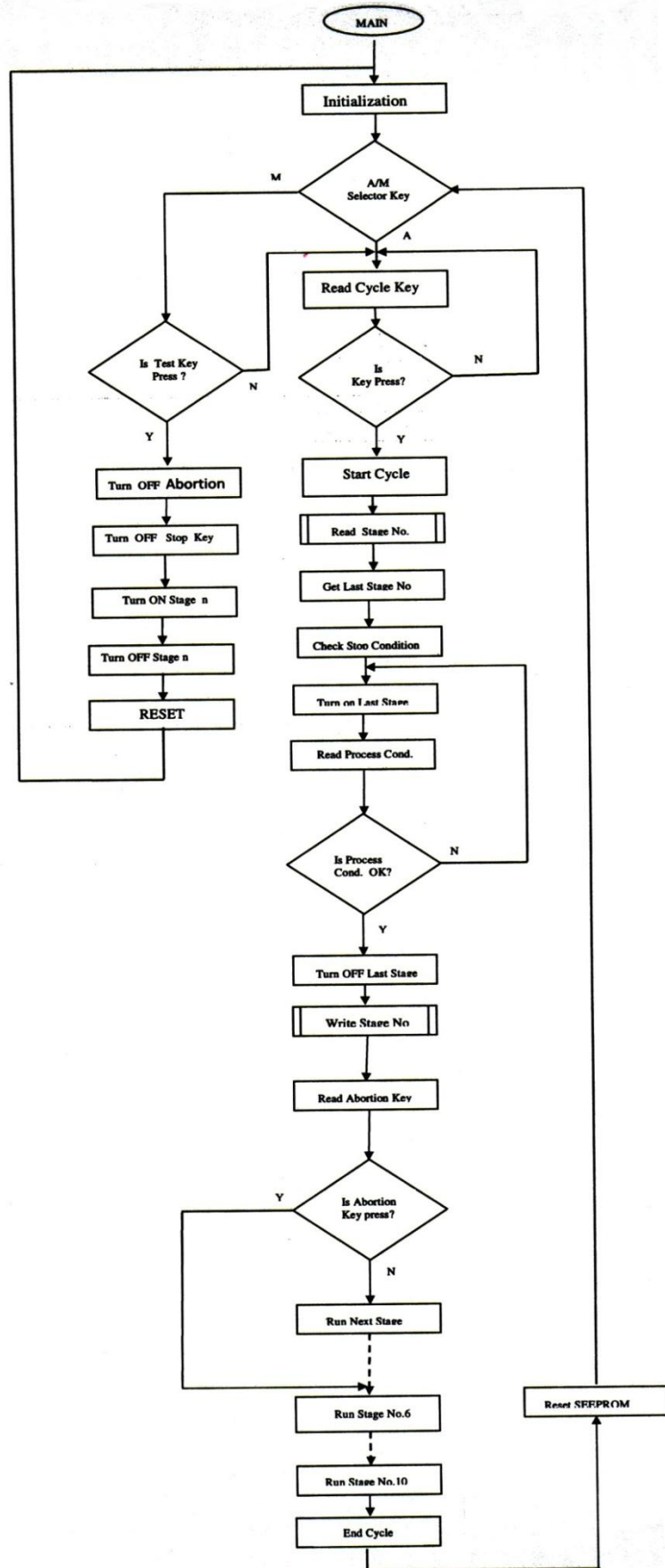


Figure 7

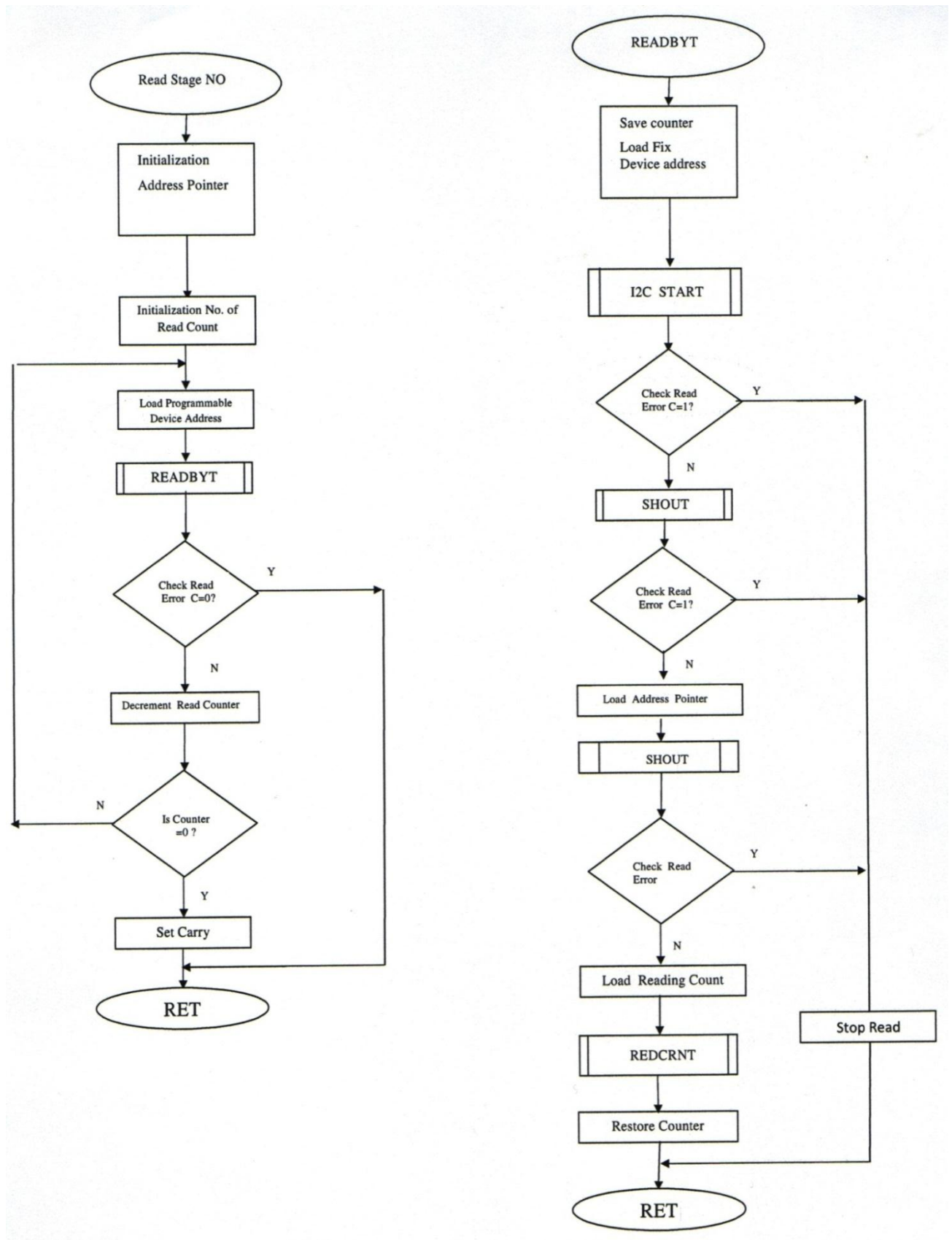


Figure 8

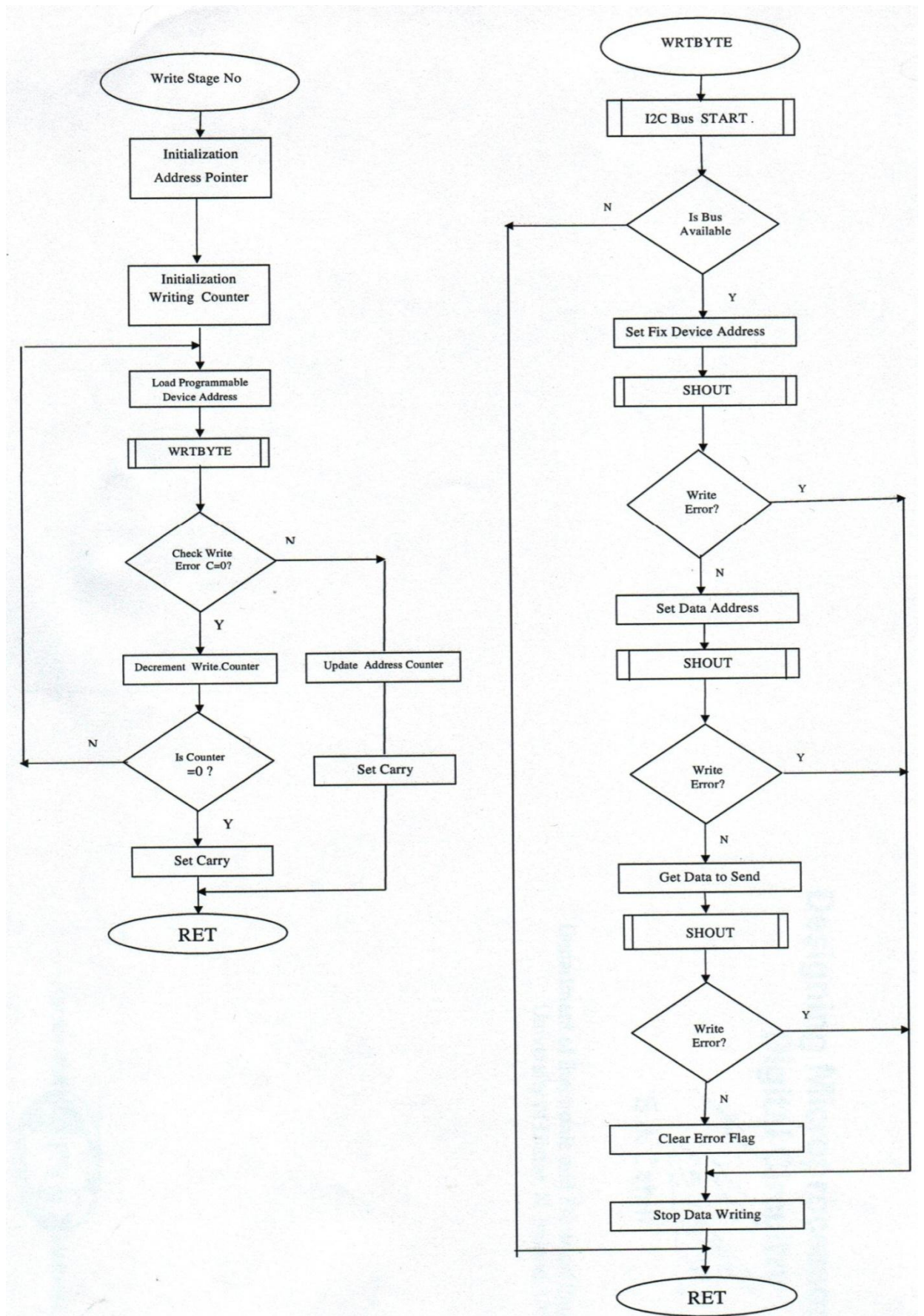


Figure 9

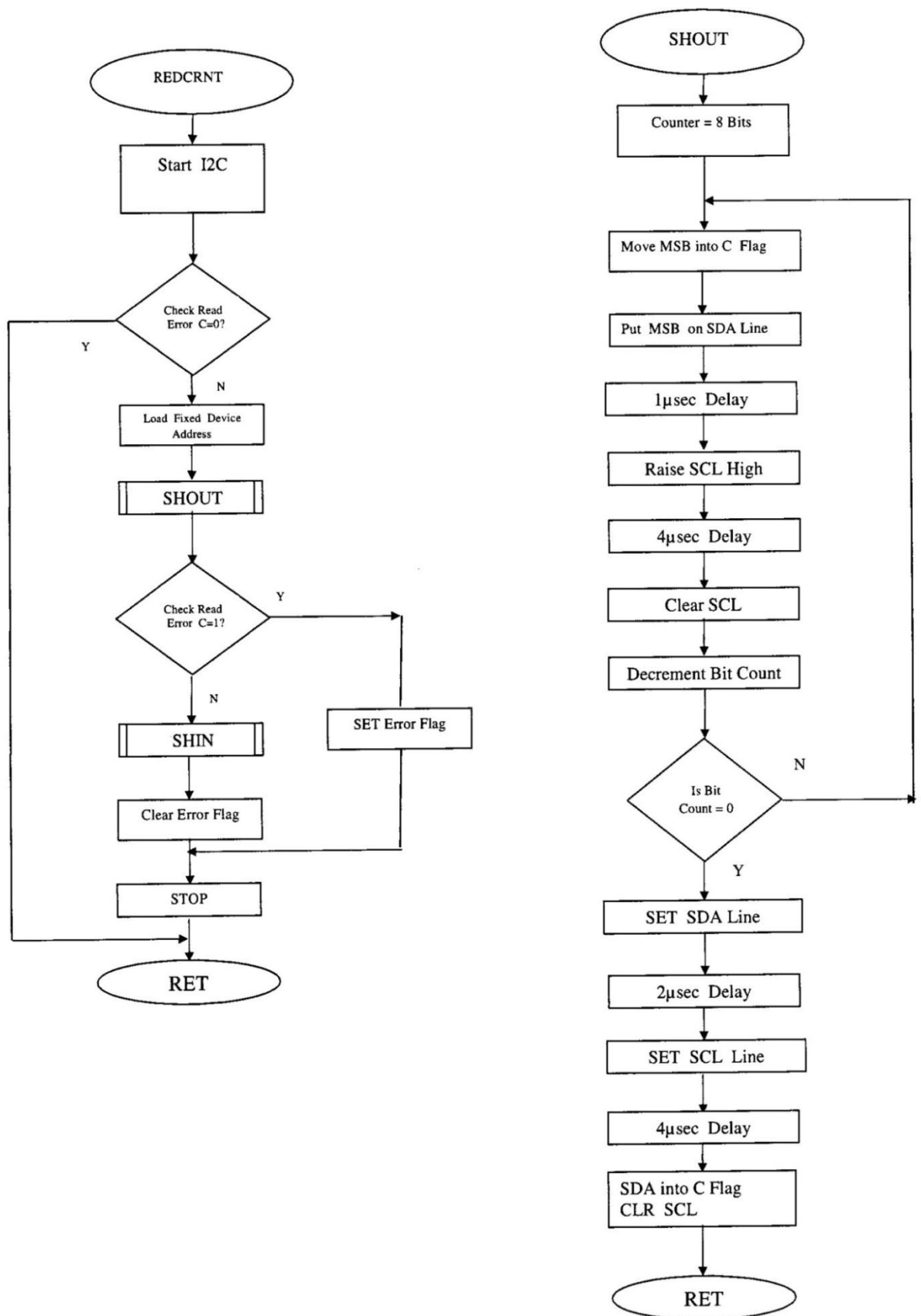


Figure 10

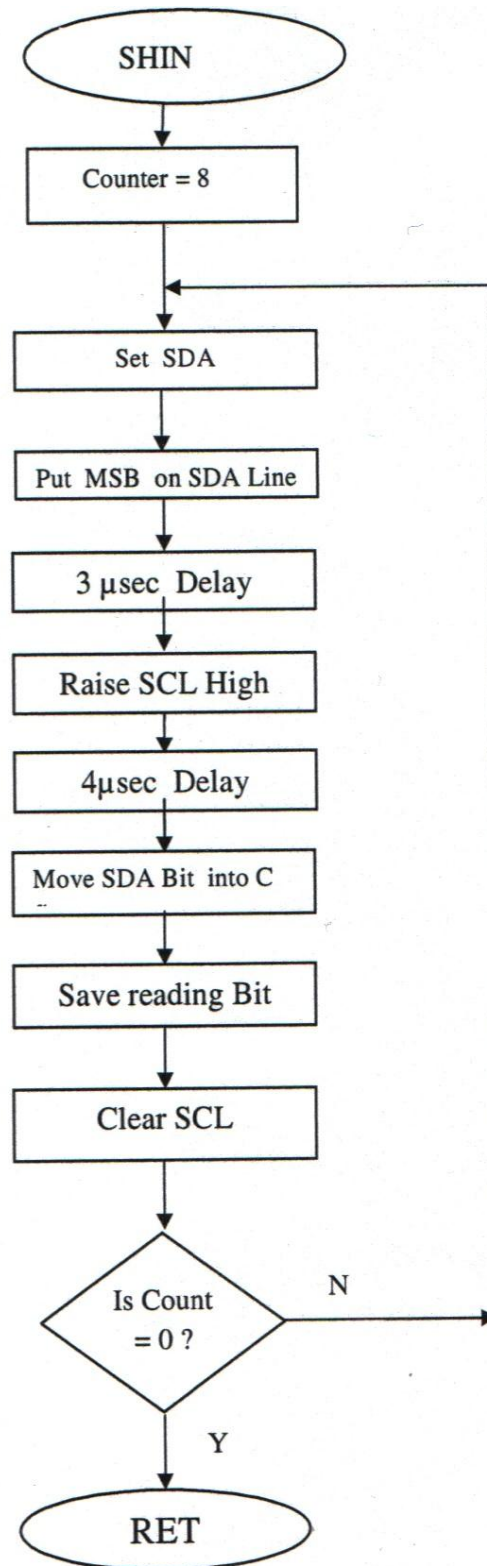
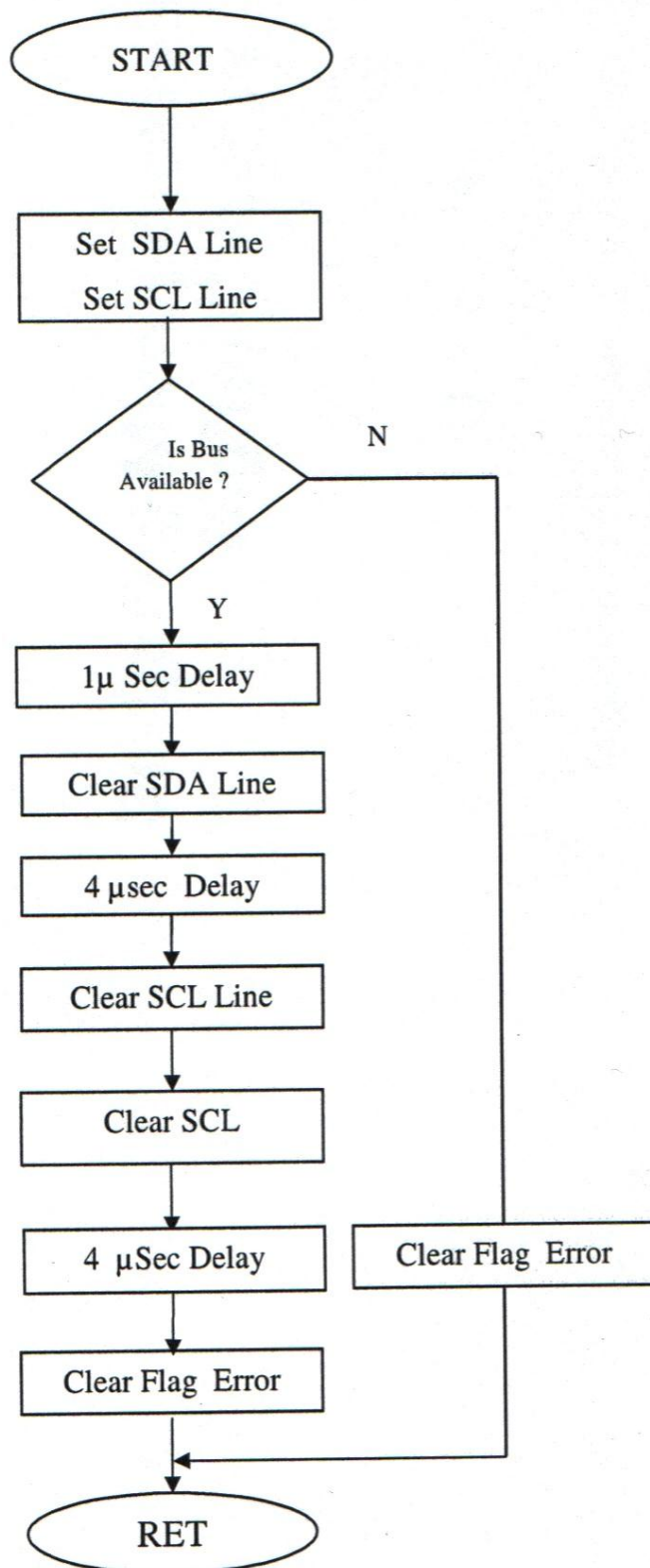


Figure 11

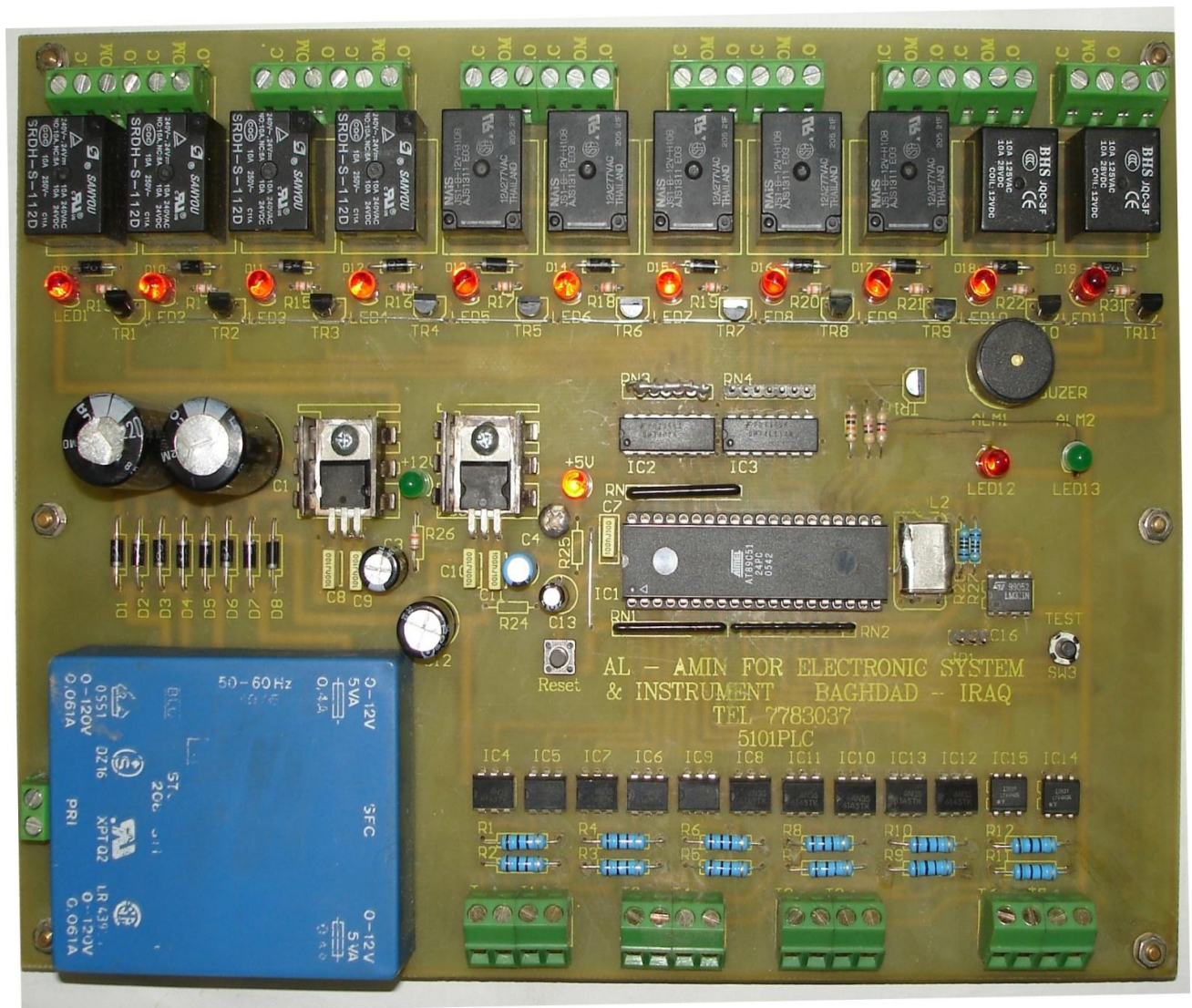


Figure 12 controller card

تصميم مسيطر تعاقبي مبرمج باستخدام ناقل I2C

رأفت صفاء الدين حبيب

قسم الهندسة الكهربائية والحاسبات

جامعة دهوك

الخلاصة

المسيطر التعاقبي من وجهة نظر هندسة السيطرة يعتبر مسيطر ذو الحلقة المفتوحة Open Loop وغالبا ما يدعى بالمتعاقب. المتعاقب يعمل بطريقة الزمن المحدد مسبقا او عن طريق حصول حدث ما خلال فترة زمنية قصيرة Discrete Event وبصورة متكررة ودقيقة. يعتمد أساس تصميم المسيطر التعاقبي التقليدي من النوع الكهروميكانيكي على استخدام الكامنة الدوارة لتحديد موضع الحدث/ الحالة State . يستخدم هذا النوع من المسيطرات في العمليات الصناعية أو الوسائل المراد السيطرة عليها ويوصف بعدد من الأحداث/الحالات أو من التوقيات المتعاقبة. بالرغم من أن عمل هذا من المتعاقبات غير مرن وغير متين بسبب استخدام أجزاء متحركة في تركيبه، لكن يبقى هذا النوع يملك ميزات حسنة تتفوق على النوع الإلكتروني الاعتيادي من كونه يحتفظ بموضع الحدث/ الحالة عندما ينقطع المصدر المجهز للطاقة الكهربائية لأي سبب كان ويستأنف المسيطر عمله مجددا بعد عودة تجهيز الطاقة الكهربائية من الموضع الذي أنقطع به المصدر الكهربائي. التقنية الحديثة مكنت من تطوير نوع من المسيطرات التعاقبية المبرمجة معتمدا على التطور الذي حدث في مجال صناعة الرقائق الإلكترونية. تحتوي هذه الرقائق التي على وحدة المعالجة المركزية CPU والذاكرة العشوائية RAM وذاكرة القراءة فقط ROM والموقتات Timer ووحدة الاتصال المتوالي UART ومنصات الإدخال والإخراج Ports متوفرة حاليا وتصنف ضمن المنظومات المضمرة mbedded System وتدعى المسيطرات المايكروية Microcontroller . المسيطرات المايكروية أعلاه تعتبر من الوسائل المبرمجة وهذا يعني ان هذه المسيطرات تنفذ البرنامج المحمل عليها بالمتعاقب من أعلى الى الأسفل (نهاية البرنامج). هذه المسيطرات لا تحفظ موضع الحدث/الحالة في حالة انقطاع مصدر الطاقة الكهربائية، بل أن المسيطر يستأنف عمله بعد عودة مصدر الطاقة الكهربائية مجددا من الموضع البدائي للعملية. يقدم هذا البحث تصميم مسيطر تعاقبي مبرمج يعتمد على المسيطرات المايكروية وباستخدام ذاكرة توالي برمجة/مسح كهربائي Serial EEPROM لحفظ موضع الحدث/الحالة. هذا النوع من الذاكرة تعشق مع المسيطر المايكروي بواسطة ناقل توالي I2C المكون من زوج من الأسلاك والذي يستخدم بروتوكول خاص بالناقل لنقل المعلومات بين المسيطر المايكروي والذاكرة المتوالي. المسيطر التعاقبي المراد تصميمه يحتوي على لوحة مفاتيح صغيرة وظيفتها إدخال توقيات الأحداث/الحالات عن طريق اختيار تلك القيم وعرضها على شاشة LCD . تم تطبيق هذا التصميم عمليا للسيطرة على عمليات تعقيم الحقن الطبية ولمدة طويلة وبدون مشاكل حقيقية تذكر.

References

1. Tamy Noergaard “Embedded system architecture A comprehensive Gide for Engineers and Programs” Newness 2005
2. Scott Mackenzie “8051 microcontroller” 1995 prentice-Hall, In
3. Atmel Corporation “8-bit Microcontroller with 8K Bytes Flash” 1999
4. Aix Maldonado “ I2C bus protocol and application” SASE Phillips 2010
5. ATMEL Corporation “2-wire Serial EEPROM” 2003
6. ATMEL Corporation “ Interfacing Serial EEPROM to microcontroller” 2001
7. Oudjida,A,S.Liacha; Benamrouche,D; Goudjil,M;Tiar,R;Ouchabane,A;”Design and Test of Integrated System in Nan scale technology” DTIS 2006,IEEE.
8. Sam Fleming; “Interfacing I2C Device to an Intel SMBUS Controller” Intel Corporation Jan 2009
9. P.Venkatesuern; Anol Kumar; Prosenijit Mandal; Dhabal and R.Nandi; “ A novel Opto-isolator technique for I2C bus for Glitch Elimination in an Industrial Environment”. International Journal of recent trend in engineering, Vol2.No8. November 2009
10. Thomas Kugelstadt; “Designing an isolated I2C Bus interfacing using Digital isolator”, analog application Journal 2011