Nutrients removal from municipal wastewater treatment plant effluent using Eichhornia Crassipes

Article · December 2009

CITATIONS
30

READS
180

4 authors, including:

S.R.M. Kutty
Universiti Teknologi PETRONAS
76 PUBLICATIONS **314** CITATIONS
SEE PROFILE

Mohamed Hasnain Isa
Universiti Teknologi Brunei
216 PUBLICATIONS **3,283** CITATIONS
SEE PROFILE

Amirhossein Malakahmad
Universiti Teknologi PETRONAS
93 PUBLICATIONS **495** CITATIONS
SEE PROFILE

Implementation of Program Debugger for Microprocessor Trainer

Khin Htar Nwe, Moe Moe Htun

***Abstract*—**Although it is possible to implement by the hardware only, adding the software would save cost and time consuming. In this paper a program debugger, a kind of DMA controller, is presented by using both the hardware and software. In the Microprocessor Trainer system, the DMA controller is used to read and write the system RAM, ROM and I/O directly and, to communicate between the keypad and display unit in the User Interface module. The PIC16F877A is used as the DMA controller to read or write the memory and I/O devices as demands by the User Interface module. The DMA module works its functions together with the User Interface module; it scans the commands via the User Interface module and carries on its tasks according to these commands. It can read and write the 16-bit address and access the 65536 memory space.

***Keywords*—**Microprocessor Trainer System, PIC16F877A, Program Debugger, User Interface Module

I. Introduction

DIRECT Memory Access (DMA) is a method of allowing data to be moved from one location to another in a computer without intervention from the central processor (CPU). It is also a fast way of transferring data within (and sometimes between) computers. The way that the DMA function is implemented varies between computer architectures. Direct Memory Access (DMA) is a method of allowing data to be moved from one location to another in a computer without intervention from the central processor (CPU). It is also a fast way of transferring data within (and sometimes between) computers. The way that the DMA function is implemented varies between computer architectures.

It is important to understand that although the CPU always releases the bus to the DMA when the DMA makes the request, this action is invisible to both applications and the operating system, except for slight changes in the amount of time the processor takes to execute instructions when the DMA is active. Subsequently, the processor must poll the peripheral, poll the registers in the DMA chip, or receive an interrupt from the peripheral to know for certain when a DMA transfer has completed.

Program instructions have used to transfer data from ports to memory or from memory to pots. For some applications, such as transferring data bytes to memory from a magnetic or optical disk, however, the data bytes are coming in from the disk faster than they can be read in with program instructions. In a case like this a dedicated hardware device called a direct memory access or DMA controller is used to manage the data transfer. The DMA controller temporarily borrows the address bus, data bus, and control bus from the microprocessor and transfers the data bytes directly from the disk controller to a series of memory locations. Because the data transfer is handled totally in hardware, it is much faster than it would be if done by program instructions. A DMA controller can also transfer data from memory to a port. Some DMA devices even can do memory-to memory transfers to implement fast block transfers.

The direct memory access controller is designed to improve the data transfer rate in systems which must transfer data from an I/O device to memory, or move a block of memory to an I/O device. It will also perform memory-to-memory block moves, or fill a block of memory with data from a single location.

Khin Htar Nwe, Lecturer, Hardware Department, Computer University (Lashio), Myanmar, the author was graduated from the University of Computer Studies, Yangon. (phone: 0949281527 ;e-mail: khinhtarnwe@gmail.com).

Moe Moe Htun, Lecturer, Department of Electrical Communication, Technological University (Hinthata), Myanmar, the author was graduated from the University of Computer Studies, Yangon. (phone: 0943033207; e-mail: dmmhtun@ gmail.com).

II. Related Work

Microcontrollers must contain at least two primary components – random access memory (RAM), and an instruction set. RAM is a type of internal logic unit that stores information temporarily. RAM contents disappear when the power is turned off. While RAM is used to hold any kind of data, some RAM is specialized, referred to as registers. The instruction set is a list of all commands and their corresponding functions. During operation, the microcontroller will step through a program (the firmware). Each valid instruction set and the matching internal hardware that differentiate one microcontroller from another.

Most microcontrollers also contain read-only memory (ROM), programmable read-only memory (PROM), or erasable programmable read-only memory (EPROM). They are used to store the firmware that tells the microcontroller how to operate. They are also used to store permanent lookup tables. Often these memories do not reside in the microcontroller; instead, they are contained in external ICs, and the instructions are fetched as the microcontroller runs. This enables quick and low-cost updates to the firmware by replacing the ROM.

Where would a microcontroller be without some way of communicating with the outside world? This job is left to input/output (I/O) port pins. The number of I/O pins per controllers varies greatly, plus each I/O pin can be programmed as an input or output (or even switch during the running of a program). The load (current draw) 18 that each pin can drive is usually low. If the output is expected to be a heavy load, then it is essential to use a driver chip or transistor buffer.

Most microcontrollers contain circuitry to generate the system clock. This square wave is the heartbeat of the microcontroller and all operations are synchronized to it. Obviously, it controls the speed at which the microcontroller functions. All that needed to complete the clock circuit would be the crystal or RC components. We can, therefore precisely select the operating speed critical to many applications.

III. Contribution

The main contribution is to design and implement the program debugger, a kind of DMA controller. In the Microprocessor Trainer System, isolated I/O system is used and I/O has its own processor. BIOS program cannot be written while the CPU instructions are still uncompleted. However, it is needed to test the operation condition of the I/O devices (keyboard and display unit), the system memory and the buses. The DMA controller is used for this purpose. It can read or write any address which is addressed by the CPU.

The reading and writing processes of the DMA controller are the same as the main processor. The main difference is that the main processor reads and writes according to its instructions and the DMA controller does according to the received commands via the User Interface module. The DMA controller works its tasks together with the User Interface module

IV. Hardware And Design Of The Program Debugger

A. General Description

In the Microprocessor Trainer system, the program debugger, a kind of DMA (Direct Memory Access) controller, is contained in one of the hardware modules. There are totally six modules (Processor module, Memory module, User Interface module, DMA (Direct Memory Access) module, PIO (Parallel Input/Output) module and Power Supply module) in the Trainer system and all modules are provided as separate chip, linked together via bus connections on a printed circuit board and supplied 5V DC power. The Microprocessor Trainer used the 16-bit wide address bus (A [0…15]) and data bus (D [0…15]) and four control signals such as MEMWR, MEMRD, IOWR and IORD. The system buses are used by the main processor and the DMA controller alternatively. The block diagram of the Microprocessor Trainer is shown in the Fig. 1. The program debugger required for the Microcontroller Trainer is built by both hardware and software controlled.

B. Hardware Implementation

The main parts involved in the program debugger which is a kind of DMA controller module are: the microcontroller PIC16F877A, two octal transparent latch 74ALS573Bs, two octal bus transceiver 74LS245s, one octal buffer/line driver 74LS244 and MAX-232 for RS-232C serial interfacing. The complete circuit diagram is shown in Fig. 2.

The microcontroller PIC16F877A is used as a DMA controller, two 74ALS573Bs are used to latch the address output from either the main CPU or from the DMA controller, two 74LS245s are used for asynchronous two-way communication between data buses, 74LS244 is used to drive the control bus signals and MAX-232 is used for serial communication. When the DMA controller needs to control the system buses, it will request the bus service from the main processor. The DMA controller can control the address bus; data bus and control bus as the main processor. However, the DMA controller cannot use the system buses if the main processor does not allow using it.

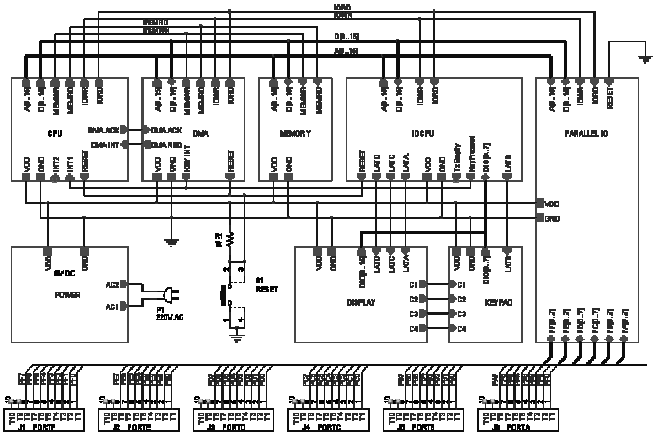

Fig. 1 Block Diagram of the Microprocessor Trainer

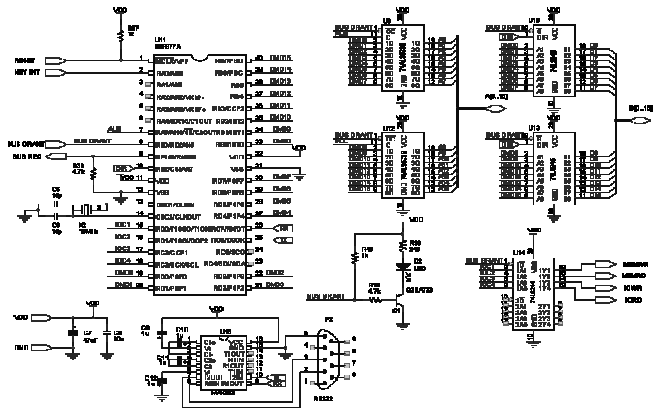

Fig. 2 Complete Circuit Diagram of the DMA Controller

C. Software Implementation

The firmware program for the microcontroller is compiled with the PIC C- Compiler Tool Suite version 3.27 from CCS C-compiler. The source is written in the MPLAB IDE version 8.0 from Microchip Corporation. To implement the actual algorithm in the PIC 16F877A, the CCS C-compiler is used with the debugger.

There are two parts of the program which are main program and interrupt program. The microcontroller will always run the main program until there is an interrupt occurred. When the microcontroller receives an interrupt flag, then it will jump to interrupt process.

1. Flow of the Main Routine

The main routine initializes port A, B, C, D and E by configuring their corresponding data direction registers with appropriate values. The voltage reference module is setup to produce 2.5 V. Its output voltage is used as a threshold voltage for the comparator module. Then it initializes comparator module to operate in mode 6.

To disable the memory and I/O modules, the control signal pins (MEMWR, MEMRD, IOWR and IORD) are being set to high level. By setting the low level for ALE (Address Latch Enable) pin, two address latch registers are prevented from transparency. Global interrupt bit and comparator module interrupt bit must be enabled for the User Interface module which can interrupt the DMA module. Since the main loop is not used for communication and other procedures, its task is to reset the watch dog timer. And the control signals such as MEMWR, MEMRD, IOWR and IORD are always being set to high level in the main loop.

2. Interrupt Service Routine

When the User Interface module wants to do one of its functions, it interrupts the DMA module via the voltage comparator module. The task of the interrupt routine is to receive the data and perform the task according to the received command. Before reading the data and commands, it requests the bus service and waits until receiving the bus grant signal. If the bus grant signal isn't received, the watch dog timer in the main loop is reset until receiving it. The User Interface module sends out the data and addresses before it gives the commands to the DMA module. After these data had been received, the commands which are needed to perform the User Interface module are given out to the DMA module. The DMA controller receives and stores the address LSB, address MSB, data LSB and data MSB. Then it performs the functions according to the commands given by the User Interface module.

3. Memory Write Procedure

Before starting memory write operation, the data bus transceiver is setup to get the data output direction. Not to cause the bus contention, it sets the control signals (MEMWR, MEMRD, IOWR, and IORD) to high level. Since the data and address are multiplexed, port B and port D send out the first address values. These address values are latched in two octal latch 74ALS573Bs. And then output the data from port B and port D and gives the inverted pulse to the MEMWR signal. In the various kinds of memory, while RAM is fast to use in writing process, EEPROM must have to take the time for the writing process. Therefore, after the memory write operation, it waits 5 ms before returning the interrupt routine.

4. Memory Read Procedure

Firstly setup the data bus transceiver to get the data input direction before the memory read operation. By putting the control signals to high level, it prevents the bus contention. Since the data and address are multiplexed, port B and port D send out the addresses and two octal latch 74ALS573Bs latch these addresses. The MEMRD pin is set to low level to read the data and it waits 100μs for data reading process. Then it reads the required data through port B and port D and sets the MEMRS pin to low level again and returns the interrupt routine.

5. IO Write Procedure

This procedure is the same as the memory write procedure.

6. IO Read Procedure

This procedure is the same as the memory read procedure.

7. Data and Address Sending Routine

This is a routine in which read data and received address are sent back to the User Interface module when memory and I/O devices are being read. In sending like this, data and address can be sent 8 bits at a time and sent totally four times. The address LSB, address MSB, data LSB and data MSB are sent back to the User Interface module. Between each interval for sending data, it must wait 100μs for the User Interface module to process the data.

D. DMA Operation

The reading or writing procedure of the DMA controller is the same as of the main CPU. The only difference is that the main CPU reads or writes according to its instructions while the DMA controller reads or writes according to the command given by the User Interface module.

Read the values and commands of data and address via the User Inter face module and then it has to read and write the memory and I/O according to the received commands. The commands from the User Interface module are MEMWR, MEMRD, IOWR, IORD and RUN commands. When the User Interface module requests the DMA service, the DMA controller will request the bus service from the main CPU by putting the DMA REQ pin to high level.

The DMA controller reads or writes the data from or to the memory when the address and MEMRD or MEMRD command is sent from the User Interface module and it sends back the read data to the User Interface module. When the RUN command from the User Interface module is received, the DMA controller will give back the bus service to the main processor.

The main four DMA operations are as follows:

1. The memory write (MEMWR) procedure
2. The Input/Output write (IOWR) procedure
3. The Input/Output read (IORD) procedure
4. The memory read (MEMRD) procedure

In memory write operation, the User Interface module gives address and MEMRD command first and gives data and MWMWR command to the DMA controller. The DMA controller writes the data to the required address space and sends the written data back to the User Interface module.

In memory read operation, the User Interface module gives the address an MEMRD command to the DMA controller. The DMA controller reads the required data from the memory and sends the read data back to the User Interface module. The I/O write and I/O read operations are the same as the memory write and memory read operations.

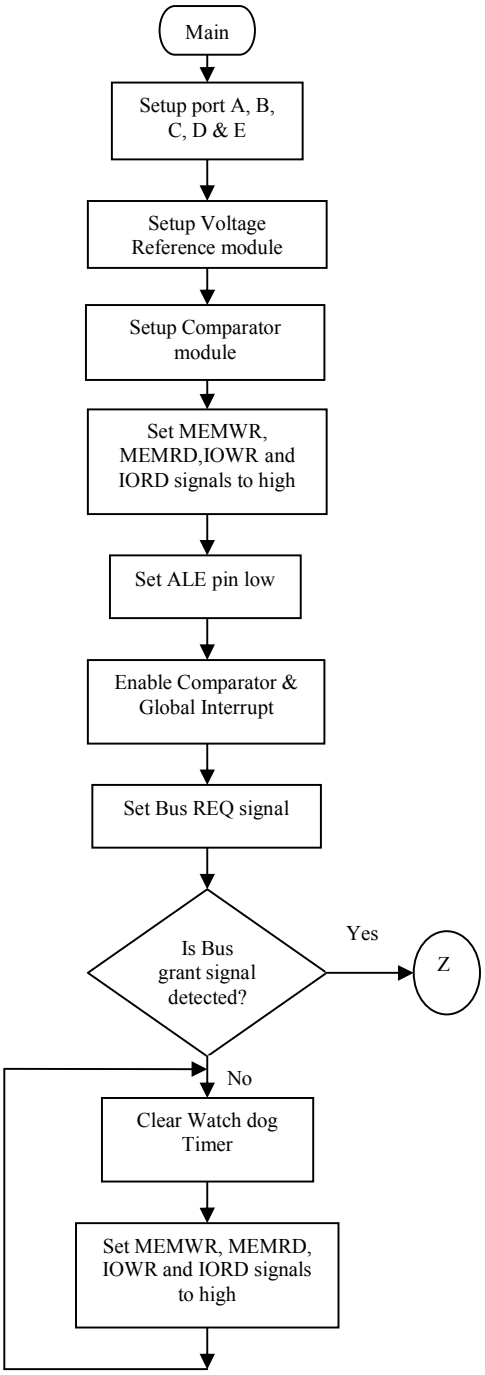

Fig. 3 Flow Chart for Main Routine

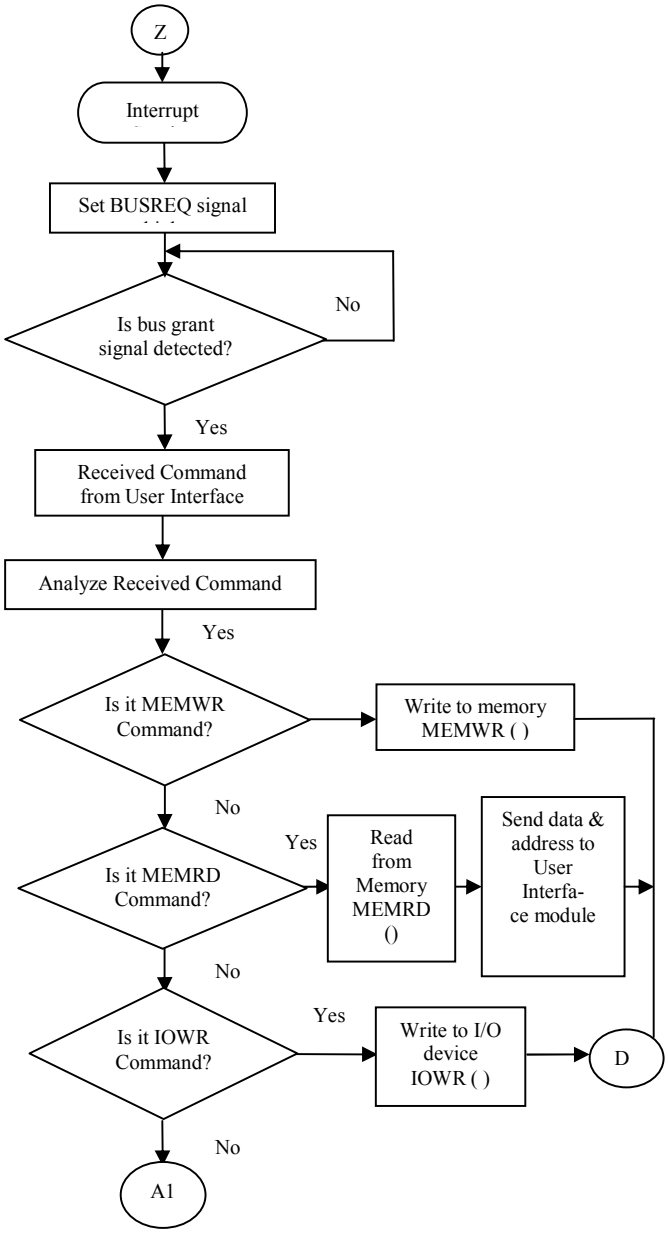

Fig. 4 (a) Flow Chart for Interrupt Service Routine

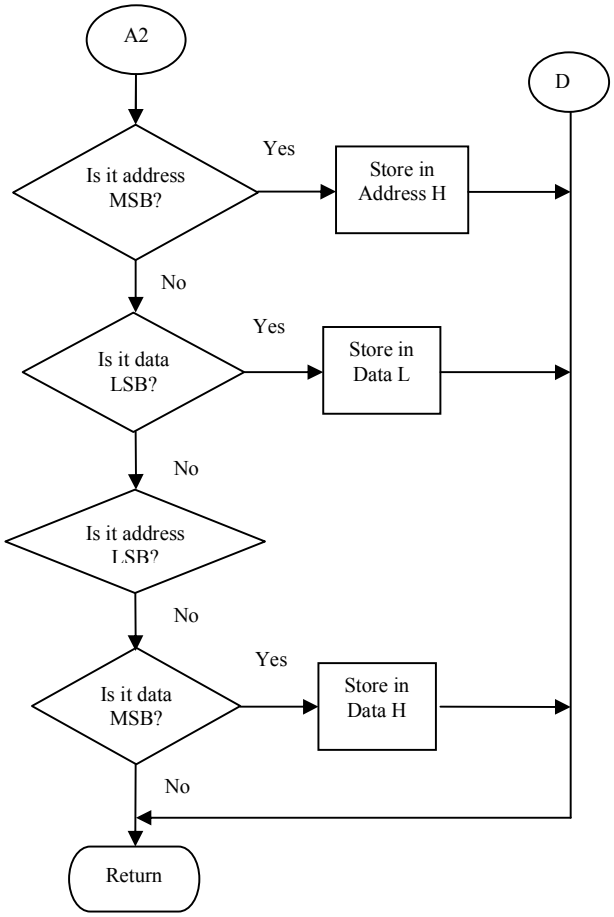

Fig. 4 (b) Flow Chart for Interrupt Service Routine (Continued)

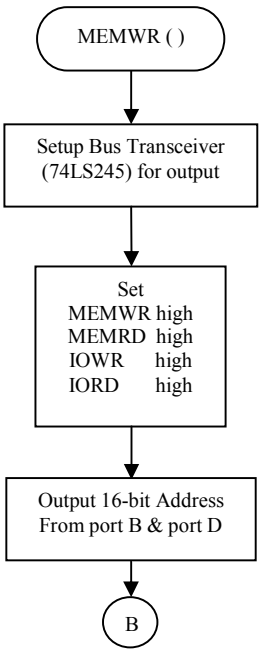

Fig. 5 (a) Flow Chart for Memory Write Procedure

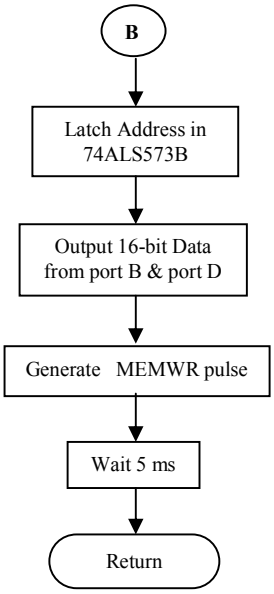

Fig. 5 (b) Flow Chart for Memory Write Procedure (Continued)

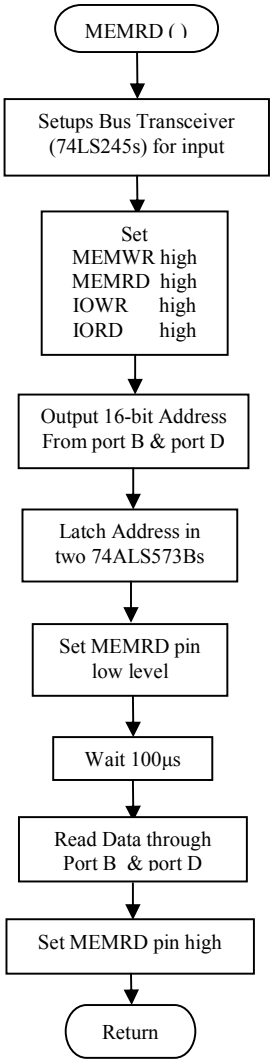

Fig. 6 Flow Chart for Memory Read Procedure

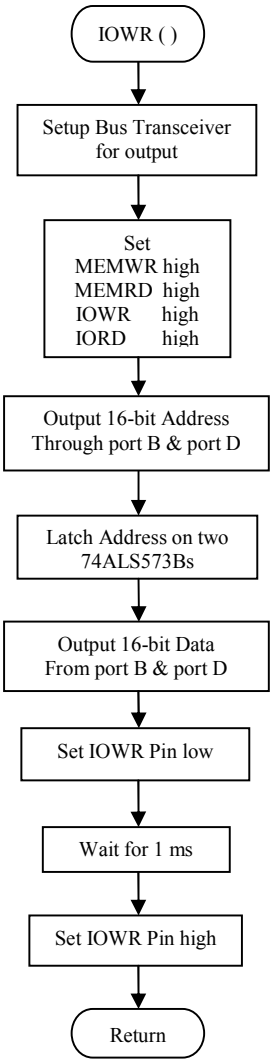

Fig. 7 Flow Chart for I/O Write Procedure

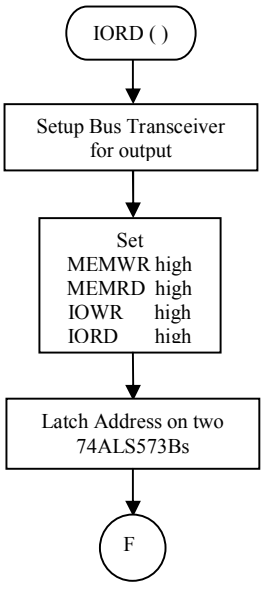

Fig. 8 (a) Flow Chart for I/O Read Procedure

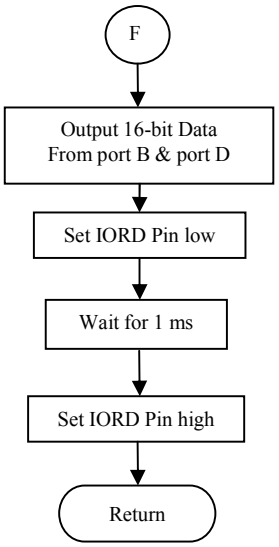

Fig. 8 (b) Flow Chart for I/O Read Procedure (Continued)

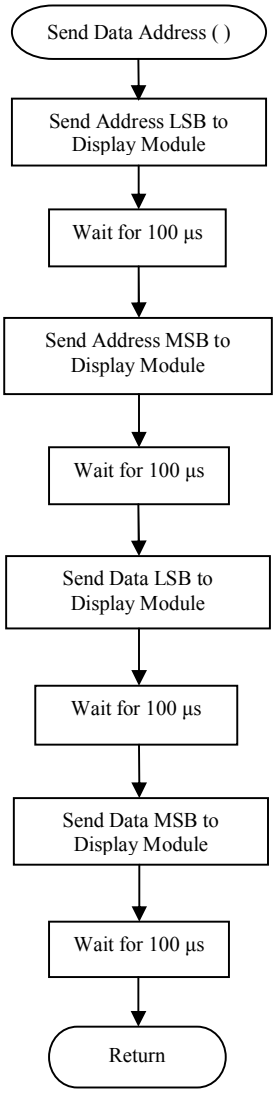

Fig. 9 Flow Chart for Data and Address Sending Routine

V. Conclusion

The aim of this paper is to design and implement a program debugger, a kind of DMA controller required for the Microprocessor Trainer. In building a microcomputer, it needs to test the memory, the Input/Output devices, the buses and the I/O ports as any program is not being written. While it is trying to develop the CPU instructions, it cannot write the BIOS program. Thus, a device is needed to do this function. A program debugger, a kind of DMA controller is used in this Trainer system for this purpose. It can read or write the memory and I/O ports. By using the DMA controller, the operation condition of Input/Output devices (keyboard and display unit) can be tested well.

If the CPU instructions are complete and its programs can be tested well, the DMA controller will not be needed so far. At that time the BIOS program can be written and instead of using the DMA controller, BIOS program will be used to test the input/output devices, memory, buses and I/O ports. However, the DMA controller is still useful in debugging the external program and in reading or writing the memory and I/O ports from externally although BIOS program have been written and used.

Although this DMA controller is designed and constructed separately from the Display Unit, it can be constructed together with the Display Unit in a module. If it does so, the design is more compact and well used. Now this design is made only for testing easily.

The problem encountered is that the direct output from microcontroller PIC 16F877A cannot get because the DMA output is limited by buffering with 74 series. If direct output from PIC is got, the DMA controller will be more useful. Instead of using the 74 series buffer 16-bit PIC can be used to get the PIC direct output. In this DMA design, since the PIC16F877A is a kind of 10-bit microcontroller the PIC outputs 8 bits only. Thus, 74 series buffer is used in this design and cannot get the PIC direct output. The microcontroller PIC 24F series can be used to get the 16-bit processor and if it can be used, the DMA controller design is better.

Acknowledgment

First of all, I am very much indebted to Dr. Ni Lar Thein, Rector of University of Computer Studies, Yangon, for her enthusiastic support and permission to carry out this paper.

I would like to profoundly grateful to U Kyaw Swa Soe, Pro-rector, University of Computer Studies, Yangon for his invaluable advice, guidance and encouragement on this paper.

Thanks are also extended to Professor Daw New Ni, Head of Computer Hardware Technology Department, University of Computer Studies, Yangon. I would like to express my gratitude and deep appreciation to Dr. Mie Mie Thet Thwin, Rector, University of Computer Studies, Mandalay for her valuable supporting to complete the whole paper.

Her Sincere thanks go to her Supervisor, Professor Dr. Win Aye, Computer University (Mandalay) for her invaluable guidance, advance, and encouragement for this paper.

I am very thankful to U Win Khaine Moe, Deputy Director General, Myanmar Science and Technological Research Department for his kind advice and guidance to complete this paper.

I also wish to express my deepest gratitude my parents, my elder sisters and my younger brother for their encouragement, understanding and support through out the period of this painstaking research.

References

[1] David, M. C., Frederick J Cowan and G Hassan Parchizadeh, University of Portsmouth," 8051 Microcontrollers Hardware, Software and Applications", 1998.

[2] Douglas, V. H., "Microprocessor And Interfacing Programming and Hardware", 2nd Edition .

[3] Iovine, J., "PIC Microcontroller Project Book 2nd Edition", Singapore: Mc Graw-Hill, 121 – 123; 2000.

[4] James, M. S., a principal member of the technical staff at Motorola, "HC05, M68HC05 Family -- Understanding Small Microcontroller" -- Rev 2.0.

[5] Julio, S., Minnesota State University, Mankato, Maria, P. C., South Central College, North Mankato, Minnesata, "Microcontroller Programming, The Microchip PIC®", Taylar & Francis Group, LIC, 2007.

[6] Ken. A., Embedded Technology TH Series, "Embedded Controller Hardware Design", LLH Technology Publishing, 2000.

[7] Lawrence, A. D., "The Microcontroller Beginner's Handbook, 2nd Edition", United State of America: Prompt Publication. 3-5;1998.

[8] MPLAB IDE, Simulator, Editor User's Guide.

[9] Murray, S., and Richard L. S., The University of Arizona, "The IBM Personal Computer™ From the Inside Out, Revised Edition".

[10] Sajjan G. S., "Computer Design and Architecture", Third Edition, Revised and Expanded, 2000.

[11] Sickle, T.V., "Programming Microcontrollers in C", LLH Technology Publishing, Second Edition, 2001.

[12] Frank, D., "FreeBSD Developer's Handbook", 8 October 1997.