EE 3232 DIGITAL SYSTEMS III
6.1 INTRODUCTION TO THE SBC188 COMPUTER SYSTEM

OBJECTIVES

January 2000 CPD

To become familiar with the operation of the basic SBC188 computer by : examining the system hardware, developing and compiling “C” programs, and using the Debug/Monitor to load, debug, execute and verify the operation of a variety of "C" programs. Another objective is to develop the software interface for using incorporating the following peripherals into an application: an 82C55A parallel peripheral interface for implementing simple parallel IO ports and the built-in programmable timers of the 80C188XL for measuring the duration of pulses generated by an IR Remote transmitter. These software interfaces will then be incorporated into a system that detects and decodes key presses entered at an IR remote transmitter.

REFERENCES

1. “The SBC188 User’s Manual”, available in the lab.
2. “Paradigm Locate Reference Manual”, available in the lab.
3. “Paradigm Debug/RT-186 Users Guide”, available in the lab.
4. “The 80C186XL/80C188XL Users Manual”, available in the lab. Also at http://developer.intel.com/design/intarch/manuals/
5. “Embedded Microprocessor System Design”, K.L. Short, Sections 4.9, 4.10, 4.11. Paradigm Tools.
6. “Embedded Microprocessor System Design”, K.L. Short, Sections 12.10. The 82C55A Paralle Peripheral Interface or PPI.
7. “Embedded Microprocessor System Design”, K.L. Short, Sections 13.7. The 80C188 Internal Timers.
8. “Embedded Microprocessor System Design”, K.L. Short, Sections 20.7. Combining “C” and Assembly Language Programming.

EQUIPMENT

1. SBC188 Computer and Host PC.
2. IO Interface Board.
3. One 26-pin cable.
4. Three10-pin cables.
5. IR receiver module.
6. Sony IR remote transmitter (with batteries).
7. Development software located in L:\EE3232\LMN

BACKGROUND

SBC188 Development System Components

The development system consists of a host computer with access to the development software

(available at L:\EE3232\LMN), the SBC188 single board computer and the IO board shown in Figure 6.1.1. “C” source programs will be developed on the host, then compiled, linked and located using the batch command files MC.BAT or MCL.BAT available in the course subdirectory L:\EE3232. The batch command file accepts a “C” source program and creates a number of intermediate files. One of the files created is an executable or .AXE file that may be downloaded into the SBC188 and executed.

The .AXE file must first be loaded into the (RAM) memory on the SBC188. To load the .AXE file use the following steps.

1. Power up the SBC188 or press the SBC188 RESET button : This causes the PDRemote Debug/Monitor program in the EPROM on the SBC188 to execute.

2. From a DOS prompt run the program PDRT186 on the host.

The PDRemote Debug/Monitor allows users to control the SBC188 microcomputer from a host computer connected to the serial port. The PDRemote Debug/Monitor program resides in EPROM on the SBC188 board and communicates with the host PC via the 10 pin serial cable connected to J202. The monitor allows you to enter programs and data into memory, execute programs, examine and change the contents of memory and registers and perform a variety of other operations useful for debugging. Refer to Section 3.5 for a summary of the Debug/Monitor commands.

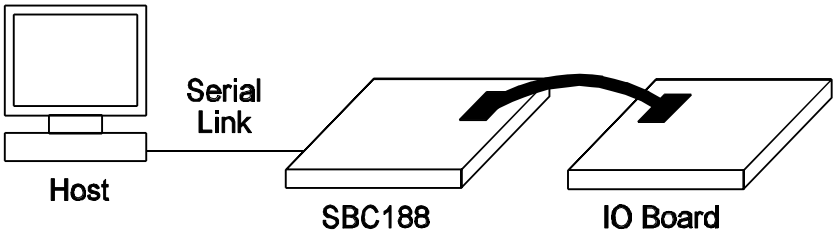

Figure 6.1.1. The SBC188 Development System

Data Representation

Data entered and displayed via the Debug/Monitor is normally in decimal or hexadecimal format listed below.

Decimal	Hex Representation		Binary
5	0x05	05H	0000 0101
9	0x09	09H	0000 1001
12	0x0C	0CH	0000 1100
15	0x0F	0FH	0000 1111
23	0x17	17H	0001 0111
79	0x4F	4FH	0100 1111
126	0x7E	7EH	0111 1110

Table 6.1.1. Data Representation Formats.

Resetting the Computer

The SBC188 microcomputer is automatically reset when powered up. To manually reset the microcomputer press the **reset** button on the edge of the SBC188 printed circuit board. When reset, the SBC188 computer automatically executes the PDRemote Debug/Monitor program that resides in EPROM.

PREPARATION

Become familiar with the contents of your laboratory manual. It contains most of the information regarding hardware and software that you will require to complete all experiments. Read Section 3, 'Software Development Tools', Section 4 on 'Microcomputer Hardware Troubleshooting' and Section 5, “C” Programming and Examples.

One of the objectives of the lab is to become familiar with the development tools for creating “C” programs and then debugging program operation on the SBC188's. To illustrate the development process we will start with two simple programs. The first interfaces with a PPI and the second, a hardware timer. These programs will form the starting point for developing an application that detects a key press at an IR Remote Transmitter and then encodes the key press.

The software that implements the IR receiver provides a wireless link between th IR Remote Transmitter and the SBC188 microcomputer. The Remote may be integrated into a variety of interesting applications. For example one could incorporate the IR Remote with the stepper motor control system visited in EE3221 and use the remote to control speed, direction, number of steps and step size for example!

The 82C55A Parallel Peripheral Interface or PPI

Read Section 12.10 in Short and Chapter 5 in your course notes to become familiar with the operating modes available with the 82C55A PPI. There are two 82C55A’s interfaced to the SBC188. Consult the schematics Sheet 1/11, 5/11 and 6/11 in Sections 7.1, 7.5 and 7.6 for details of the system bus interface. You will use the 82C55A whose base address is 0x138 for this lab. Consult the IO port map in your Lab Manual to determine the IO addresses where the ports denoted by KEY_PA, KEY_PB, KEY_PC and the Control Register are mapped. You should be able to determine that the control word necessary to program 82C55A operation with PA = mode-0 output, PB = mode-0 input, PCL = mode-0 output and PCH = mode-0 output, is 0x82. You will need to initialize the 82C55A by writing 0x82 to the Control Register within the 82C52A as part of your initialization code.

The program L:\EE3232\L311.C is an example of a “C” program that initializes the 82C55A and then waits for a 0 to 1 transition at bit-0 of input port KEY_PB. The program L:\EE3232\L312.C is an example of a “C” program that performs the same operations as L311.C except the program is modularized using function calls.

The 80C188XL Internal Timer 1

Read Section 13.7 of Short and Chapter 10 of your class notes for additional information on timer operation. The 80C188XL has three binary up-counters, **Timer0**, **Timer1**, **Timer2**. The programmer may write a control word to the 16-bit port, TxCON, to program the mode of operation and to enable/disable counting of the timers, x = 0, 1 or 2. The timer "count values" may be read or written "on-the-fly" at the 16-bit port, TxCNT.

Timer 2 is clocked internally at 0.25*CPU clock frequency = 4 MHz and is used primarily as a pre-scaler for Timers 0 and 1. In this lab we will program Timer 2 to count up to 4 before resetting, i.e., the time out value is 4 ÷ 4 MHz = 1 microsecond. Every time Timer 2 "times out" it triggers or clocks Timer 1. This causes Timer 1 to count up by 1 every microsecond.

The program L:\EE3232\L313.C is an example of a "C" program that initializes Timer 2 and Timer 1 such that Timer 1 counts up from an initial value of 0x0000 once every microsecond. Every time the count (read from Timer 1) reaches 50000, (i.e., 0.05 milliseconds has elapsed), the program variable, C, is incremented and written to the 8-bit output port, KEY_PC.

Read Appendix 6.1.1 attached to this document to become familiar with the operation of an IR Remote Transmitter and Receiver. In this lab you will develop and debug a series of program modules that will decode the key presses transmitted by the Sony IR Transmitters.

One way of decoding the pulse length modulated signal of the IR transmitter is to first develop the following procedures.

> **void WAITN(void)** : waits until a transition from '1' to '0' is detected at bit-0 of port PB.
> **void WAITP(void)** : waits until a transition from '0' to '1' is detected at bit-0 of port PB.
> **void TIMER_INIT(void)** : initializes the 80C188XL Timer 1, to count (up) from 0x0000 with a period of 1 microsecond.

The procedures WAITP() and TIMER_INIT() already exist in L312.C and L313.C. You may copy these from L:\EE3232 and use them later in the lab!

The ***time*** in units of microseconds may be read at the timer port, T1CNT, using **time = inport(T1CNT)** where **time** is the elapsed time in microseconds since **TIMER_INIT()** was last executed! The next step is to design a procedure to wait for a valid start bit and then read in the 12-bits of IR data and return as an unsigned word. The procedure must first wait until a start bit is detected. One way of doing this is to measure the time between a negative and positive transition at bit-0 of port, KEY_PB, and repeat this operation until the measured time is 1900 < **time** < 2800. Then read in the 12-bits of data. Each data bit may be detected in a similar manner. Measure the time between a negative and positive transition at bit-0 of port, KEY_PB. If the measured time is **time** < 900 then the data bit = '0' else the data bit = '1'.

Consider a function, ***unsigned DURATION_LOW(void)***, that waits for a 1 to 0 transition at bit-0 of input port KEY_PB, starts Timer 1 counting up in units of microseconds, waits for a transition from 0 to 1 at bit-0 of port KEY_PB, reads the count value at Timer 1 and then returns this as an unsigned value to the calling program as follows.

```
unsigned DURATION_LOW(void)
{
WAITN() ;              // Wait for a 1 to 0 transition at bit-0 of port KEY_PB.
TIMER_INIT() ;         // Start Timer 1 counting up in units of microseconds.
WAITP() ;              // Wait for a 0 to 1 transition at bit-0 of port KEY_PB.
return(inport(T1CNT)) ;
}
```

This function is particularly useful because it can be used to determine the duration of the low pulses generated by each key press of the IR Transmitter. By measuring all of the LO times and sorting through the assignment of 1's and 0's one may decode the key presses of the IR transmitter by regenerating the encoded values of the key presses in Table 6.1.2. You may use the functions in L312.C and L313.C as templates for implementing DURATION_LOW().

1. Using function, DURATION_LOW(), write a program that measures the duration of the start bit, TS, in microseconds and then writes the 16-bit value to the 8-bit output ports KEY_PC and KEY_PA.

2. Write a program that measures the duration of the pulse representing bit-0 in Figures 6.1.5 and 6.1.6, and then writes the 16-bit duration to the 8-bit output ports KEY_PC and KEY_PA.

3. Devise a function, ***unsigned IR_READ(void)***, that waits for a 12-bit character to be transmitted by the IR remote and returns the 12-bit encoded value. The algorithm may be implemented by waiting until DURATION_LOW() returns a value between 1900 and 2800 signaling a start bit! Then the algorithm may call DURATION_LOW() twelve more times. Each time measuring the duration of the low pulse for the next data bit and assigning a logic-0 if the measured time is < 900 or a logic-1 if the measured time is > 900.

EXPERIMENT

1. Connect a 26-pin cable from J602 on the SBC188 to the IO Interface board. Be sure to properly align the connector to the header. Connect a 10-pin cable from port KEY_PC to the LED's and from port KEY_PA to the LED's on the interface board. Also connect a 10-pin cable from port KEY_PB to the switches on the interface board. NOTE : the two most significant bits of KEY_PB are always read as “1”.

2. Identify the location, device name where applicable, and the function(s) of the following devices on the SBC188.

   80C188XL (label U101)
   TL16C452 (label U201)
   8255A (label U501 and 602)
   U401
   U406
   82C206 (label UA01)

Refer to Section 2 and the schematics of the SBC188 in the SBC188 Users Manual available in the lab.

3. **Using the Software Development Tools for .C Source Programs**

Invoke at the command line **mc L311.C**. mc is a batch program that will automatically compile, link and locate the .C source program creating the following files,

tmp.c	Copy of the original source program.
tmp.obj	Compiler output file.
tmp.asm	Optional compiler output in equivalent assembly code.
tmp.map	Linker map file - you may look at this.
tmp.rom	Linker relocatable file.
tmp.cfg	Locate configuration file for the SBC188.
tmp.loc	Map file generated by locate.
tmp.exe	Executable for running/debugging on a DOS machine.
tmp.axe	Executable for running on the SBC188.

You might like to examine the .ASM, .MAP, .LST, .CFG and .LOC files with a text editor.

Ensure the serial cable is connected between J202 of the SBC188 and the serial port of the host PC. Turn on the power to the SBC188 then run PDRT186 on your host PC. Your host PC should have the standard DEBUG/MONITOR display on the screen. Section 3.5 explains some of the functionality of the various menus of the display.

You are now ready to debug the program using the Debug/Monitor.

(a) We will first test operation of the IO ports. Make sure the 26-pin cable is connected between the 82C55A at J602 and the IO Board.

Go to the View-CPU menu. Press Alt-F10 or hit the right mouse button while inside the CPU window. Click on IO. Select Out Byte. Write 0x82 to port 0x013E to configure the 82C55A with ports, KEY_PA as output, KEY_PB as input and KEY_PC as output. Now write 0x55 to output port KEY_PC (address 0x013C) and port KEY_PA (address 0x138) and verify the correct pattern on the LED's. Repeat with 0xAA. Now apply a binary pattern to input port KEY_PB via the switches. Use the IO In Byte command to read KEY_PB (address 0x13A) and verify that the correct pattern is read. Repeat for different patterns. Now exit the CPU window.

(b) Go to the Data-Add Watch menu and add the variables, X and Y. What is the initial value of the Watch variables?

(c) Set a breakpoint at the first "while" statement by clicking on the dot in the left most column of the while statement. The instruction should now be highlighted. Go to the Run menu and execute run. What is the value of the watch variable. Resume execution from the breakpoint by going to the run menu and clicking run or hitting the F9 key. The program should terminate gracefully!

(d) Go to the Run Menu and hit Program Reset. Now trace through each instruction, one at a time by clicking on Trace at the bottom of the menu or hitting F7. Now verify the logical execution of the program and observe the Watch Variables!

(e) Reset the program. Run the program until the breakpoint is reached. Select the Watch window, right-click-change, then enter a new value for watch variable(s), then resume execution with run. You can remove the breakpoint by clicking on the dot in the left most column - the instruction should no longer be highlighted!

(f) Reset the program. Clear all breakpoints. Now click Run-Animate and observe program execution!

(g) Reset the program. Run the program by pressing F9. When the breakpoint is reached select the Watch window, right-click-change, then enter a new value for watch variable(s), then resume execution with run. You can remove the breakpoint by clicking on the dot in the left most column - the instruction should no longer be highlighted!

(h) Reset the program. Open up the View-CPU menu. The CPU window appears. Right Click **Goto CS:IP**. Notice that each "C" language statement is displayed together with the compiled assembly language instructions and the associated machine code. Trace the program. Notice that control now traces through each assembly language instruction instead of each C instruction! You may also view the CPU registers and flags as each assembly language instruction is traced!

Now exit the Debug/Monitor.

4. Compile/link the following programs on the host computer using mc.

L312.C	Program the 82C55A PPI and wait for a transition at bit-0 of input port KEY_PB.
L313.C	Initialize and use the 80C188XL hardware Timer 1 in a simple application.

Now load and verify the operation of these program by tracing, executing to a breakpoint, adding “watch variables” .... etc.

5. Connect the IR receiver to the white breadboard and interface it to the SBC188 as shown in the Figure 6.1.6. If available connect an oscilloscope to the output pin, B, of the powered receiver. Direct the Sony IR Remote Control at the receiver and press any button and observe the pattern on the scope. Verify the pulse length modulation protocol.

6. Load, run and debug the programs developed in Parts 1, 2 and 3 of the Preparation.

SUMMARY

Your summary should address the following,

(a) What are the main components of the SBC188 system?
(b) What is the function of RESET?
(c) What is the function of the Debug/Monitor?
(d) What is the function of the host PC?
(e) List the steps that comprise the compiling process in mc.bat.
(f) For the IR remote transmitter what is the worst case time for transmitting a single character?
(g) Submit a software listing for the IR Remote Decoder software signed and dated by the authors. Undocumented and/or unsigned source code will receive a grade of 0.

APPENDIX 6.1.2 IR Remote Transmitter and Receiver Operation

An infrared remote transmitter generates digital data one bit at a time using the infrared emitter circuit shown in the Figure 6.1.2.

By turning the control transistor on and off one can transmit logic '1's and '0's as the presence or absence of IR radiation. Most IR remote controls transmit information by modulating the serial data with a 40 KHz carrier to reduce the effect of ambient radiation (mostly caused by fluorescent lights) on the transmitted IR signal. Therefore a logic '1' is transmitted as a burst of IR radiation at 40 KHz and a logic '0' is transmitted as an absence of IR radiation as shown in the Figure 6.1.3.

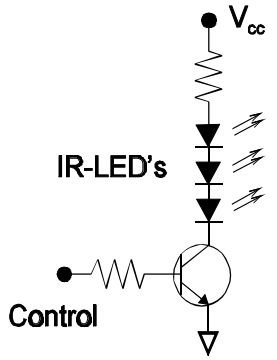

Figure 6.1.2. IR Emitter.

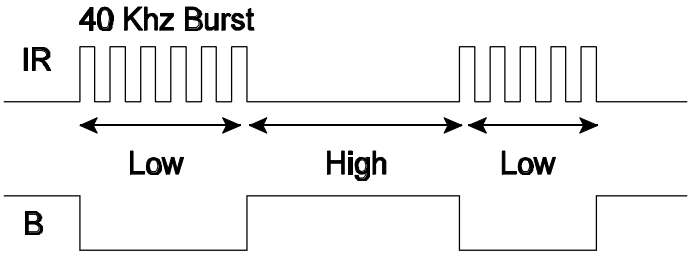

Figure 6.1.3.

In this lab you will use a commercial remote control unit to transmit IR serial data which is received by the ***IR receiver module*** and you will interface the IR receiver module to the SBC188 microcomputer as shown in Figure 6.1.4 so that software may be written to detect and decode the digital data transmitted by the IR Remote. The IR receive module detects the presence of 40 KHz bursts of IR radiation and generates (demodulates) a logic output, B, as shown in Figures 6.1.3 and 6.1.4.

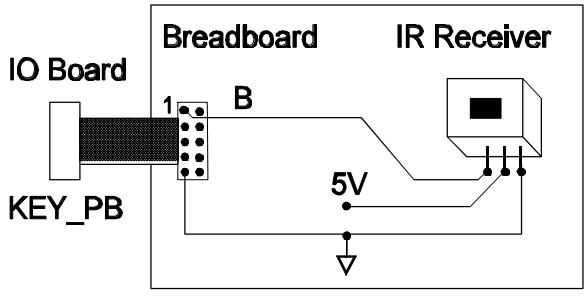

Figure 6.1.4. The IR Receiver Interface.

The operation of an IR receiver involves detecting the presence of a 40KHz burst and representing the entire burst as a logic 'high'. In between bursts a logic 'Low' is represented. The circuitry for demodulating the logic values from the 40 KHz carrier is similar to the way signals are demodulated in an AM radio receiver!

In this experiment you will interface an IR receiver to the SBC188. The receiver has internal circuitry to demodulate the 40 KHz IR signal and generate a logic output that can be directly interfaced to a input port.

One of the common standards for encoding information in IR transmissions uses pulse length modulation. Each 12-bit character is transmitted using asynchronous serial communication. Initially the line is in the idle state = '1'. A start bit, S, = '0' of duration TS = 2.4 ms signifies the beginning of a transmission. Each transmitted data bit is encoded by: a high level of duration, T = 0.52 ms, followed by a low level of duration, T0 = 0.68 ms, (representing logic '0') or a high level of duration, T = 0.52 ms, followed by a low level of duration, T1 = 1.25 ms, (representing logic '1'). Notice that a logic '1' takes more time to transmit than a logic '0'.

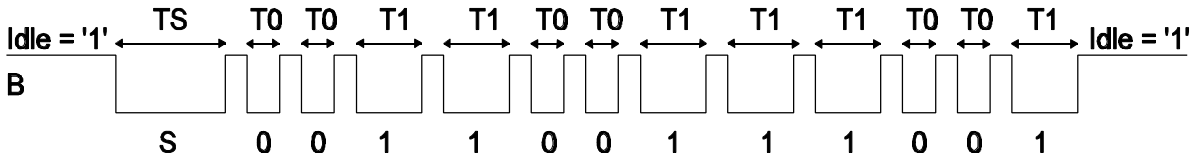

Figure 6.1.5. IR Remote Transmitter Serial Format.

Information is encoded in 13-bits beginning with the start bit, S. The start bit is used to synchronize the reading of the remaining 12-data bits, $A_0\,A_1\,A_2\,A_3\,A_4\,A_5\,A_6\,A_7\,A_8\,A_9\,A_{10}\,A_{11}$.

S	A 0	A 1	A 2	A 3	A 4	A 5	A 6	A 7	A 8	A 9	A1 0	A1 1
0	0	0	1	1	0	0	1	1	1	0	0	1
S	LSB					12-Bit Data						MSB

Figure 6.1.6. Sony IR Remote Bit Format..

12-bit Encoding for the SONY IR Remotes

The encoding of the keys for the SONY IR remotes are listed below.

KEY	Encoding	Key	Encoding	Key	Encoding
Muting	0x094	7	0x086	Ch +	0x090
Sleep	0x0B9	8	0x087	Ch -	0x091
Power	0x095	9	0x088	Jump	0x0BB
1	0x080	0	0x089	Reset	0x096
2	0x081	Display	0x0BA	(R)+	0x0F4
3	0x082	Enter	0x08B	(L)-	0x0F5
4	0x083	Ch Guide	0x08E	Menu	0x0E0
5	0x084	Vol +	0x092	Return	0x0E5
6	0x085	Vol -	0x093		

Table 6.1.2. Sony IR Remote Transmitter Encoding Table.

APPENDIX 6.1.2 Using the Monitor to Enter Data and .ASM Programs

Power up the SBC188. On the host computer run PDRT186. Connect a 26-pin cable from J602 on the SBC188 to the IO Interface board. Be sure to properly align the connector to the header. Connect a 10-pin cable from port KEY_PC to the LED's and from port KEY_PA to the LED's on the interface board. Also connect a 10-pin cable from port KEY_PB to the switches on the interface board. NOTE: the two most significant bits of KEY_PB are always read as “1”.

Examine the contents of memory location 0000:4000H. To do this: click on View-CPU to bring up the CPU window. Left click on the Data Pane in the bottom left (to select it), right click-select **Goto** and enter 0:4000. Now examine the contents of memory location 0000:4000H. Is this memory location in volatile or non-volatile memory? What about memory location FFFF:0000.

Enter the machine code for the following program one byte at a time into consecutive memory locations starting at location 0000:3FFEH. Simply **Goto** 0:3FFEH in the Data Pane, click on the byte at 0:3FFE and enter the bytes in the format xyh or 0xyH if x is not a numeral. Then select the **register pane** and initialize registers DS, AX and BX to zero.

ADDRESS	MACHINE CODE	INSTRUCTION MNEMONIC	COMMENTS
0000:3FFE	00 00	Reserved RAM	; Reserved RAM for 16-bit
			; variable INFO.
0000:4000	BB FE 3F	mov bx, 3FFEH	; Store C234H into
0000:4003	C7 07 34 C2	mov [bx], 0C234H	; variable INFO.
0000:4007	8B 07	mov ax, [bx]	; Load INFO into ax.
0000:4009	D1 C0	rol ax, 1	; Rotate ax left.
0000:400B	D1 C0	rol ax, 1	; Rotate ax left.
			;
0000:400D	25 7F 00	and ax, 7FH	; Gracefully end
0000:4010	50	push ax	; program and exit
0000:4011	BB 00 00	mov bx, 0	; to the
0000:4014	8E C3	mov es, bx	; Debug/Momitor.
0000:4016	9C	pushf	;
0000:4017	53	push bx	;
0000:4018	53	push bx	;
0000:4019	26 FF 2E 0C 00	jmp far [es:000CH]	

Select the **register pane** by pointing and left clicking the mouse. Initialize register CS = 0000h and the instruction pointer register, IP = 4000H, by right clicking on the display. Select the **Code Pane** and **Goto** CS:IP. Notice that the monitor automatically disassembles the machine code and displays instruction mnemonics! Now **trace** through the program one instruction at a time by hitting F7 or clicking on TRACE at the bottom of the screen. Stop tracing when you get to the ***and ax, 7FH*** instruction. What are the final values in registers AX and BX and the 16-bit

word variable, INFO, located in memory at location 0000:3FFE?

Now ensure registers CS and DS = 0000. In the Code Pane Goto CS:4020. Now invoke the in-line assembler: **right click assemble**. Enter the ***instruction mnemonics*** (beginning with mov bx, 3FFEH) one by one beginning at location CS:4020H. Notice that the Debug/Monitor automatically converts the instruction mnemonics into machine code!

Initialize the instruction pointer register, IP = 4020H. Now trace each of the instructions until the and ax, 7FH instruction. Examine the contents of register AX, BX, memory variable INFO (location 0000:3FFEH and 0000:3FFFH) and the flag register after execution of each instruction.

Using the Software Development Tools for .ASM Source Programs

The following assembly source programs are in the L:\EE3232 network subdirectory. You should log onto the network and copy these into your personal F:\ drive for editing and modification.

LA311.ASM	Write a value to the parallel port, KEY_PC.
LA312.ASM	Write an array of values to the parallel port KEY_PC.
LA313.ASM	Read parallel port KEY_PA and write to parallel port KEY_PC.

Invoke at the command line **ma LA211.ASM**. **ma** will assemble, link and locate the .ASM source program creating the following files:

tmp.asm	Copy of the original source program.
tmp.obj	Compiler output file.
tmp.lst	List file generated by the assembler.
tmp.map	Linker map file - you may look at this.
tmp.rom	Linker relocatable file.
tmp.cfg	Locate configuration file for the SBC188.
tmp.loc	Map file generated by locate.
tmp.exe	Executeable for running/debugging on a DOS machine.
tmp.axe	Executable for running on the SBC188.

You might like to examine the .MAP, .LST, .CFG and .LOC files with a text editor. Now load the program into the Debug/Monitor by invoking,

pdrt186 tmp

You are now ready to debug the program using the DEBUG/MONITOR.

(a) We will first test operation of the IO ports. Go to the View-CPU menu. Press Alt-F10 or hit the right mouse button while inside the CPU window. Click on IO. Select Out Byte. Write 92H to port 013EH to configure the 82C55A with KEY_PA as input, and KEY_PC as output. Now write 55H to output port KEY_PC (address 013CH) and verify the correct pattern on the LED's. Repeat with 0AAH. Now apply a binary pattern to input

port KEY_PA via the switches. Use the IO In Byte command to read KEY_PA (address 0138H) and verify that the correct pattern is read. Repeat for different patterns. Now exit the CPU window.

(b) Go to the Data-Add Watch menu and add the variable, DATA. What is the initial value of the Watch variable?

(c) Set a breakpoint at the **out dx, al** instruction (that writes the contents of DATA to port KEY_PC) by clicking on the dot in the left most column. The instruction should now be highlighted. Go to the Run menu and execute run. What is the value of the watch variable. Resume execution from the breakpoint by going to the run menu and clicking run or hitting the F9 key. The program should terminate gracefully! Examine the LED's.

(d) Go to the Run Menu and hit Program Reset. Now trace through each instruction, one at a time by clicking on Trace at the bottom of the menu or hitting F7. Verify the logical execution of the program.

(e) Reset the program. Run the program until the breakpoint is reached. Select the Watch window, right-click-change, then enter a new value for watch variable(s). Then resume execution with run. You can remove the breakpoint by clicking on the dot in the left most column - the instruction should no longer be highlighted!

(f) Reset the program. Clear all breakpoints. Now click Run-Animate and observe program execution!

You have now completed a typical debugging session! Now exit the monitor and modify one or more the .ASM programs as follows,

LA311.ASM	Change the value written to the port.
LA312.ASM	Write every second element of the array to the port.
LA313.ASM	Change the condition to: wait for bits 0 and 1 to be set (and all other bits don't care).

NOTE: give the revised files a new name!

Regenerate the .AXE file using **ma**. Load and debug your revised programs!