EE 3232 DIGITAL SYSTEM III
6.3 COMMUNICATION WITH SERIAL IO DEVICES

March 2000 CPD

OBJECTIVES:

To investigate the interface between the SBC188 system bus and a serial communication channel using the TL16C452 UART peripheral. To develop a number of software modules that initialize and program the UART device and write information to the serial display unit. To use an oscilloscope to measure the voltages on a RS-232C serial line and investigate the serial protocol and baud rate. To develop a progression of software modules that read and decode packets sent by a serial mouse and a magnetic card reader.

REFERENCES

1. "Embedded Microprocessor System Design", K.L. Short, Chapter 15, pp. 540 - 563.
2. "Microprocessors and Interfacing", 2nd Edition, R.V. Hall, pp 487-502.
3. "The 80C188XL Microprocessor Users Manual", available in the lab, also at http://developer.intel.com/design/intarch/manuals/
4. Section 8.2, "The TL16C452 Integrated Peripheral" also at http://www.ti.com/sc/ (Then search TL16C452).
5. Sections 7.1 - 7.2, "Sheets 1/11 and 2/11 of the SBC188 Schematics".
6. "The SBC188 Users Manual", available in the lab.
7. "Paradigm Debug/RT - 186 Users Guide", available in the lab.
8. "Paradigm Locate Reference Manual", available in the lab.

EQUIPMENT

1. SBC188 board.
2. Host PC.
3. IO interface board.
4. 10-pin cable.
5. Serial display unit (jumpered for RS-232C).
6. A serial mouse.
7. Magnetic card reader with RS232C interface.
8. Oscilloscope.

BACKGROUND : The UART Interface

The SBC188 uses the TL16C452 peripheral device as an interface between two serial IO channels and the SBC188 system bus. The TL16C452 is a multi-function peripheral that interfaces with a Centronics parallel port and two asynchronous serial communication channels. Each serial communication channel in the 16C452 has a built-in 8250-type UART, or universal asynchronous receiver transmitter. In this lab you will program and use one of the built-in UART's. The other UART is reserved for use with the Debug Monitor! Consult sheet 1/11 and 2/11 of SBC188 schematics in Chapter 7 and identify the following:

- The 80C188XL microprocessor.
- The 74HCT573 latch for demultiplexing the address/data pins.
- The PA7104 programmable gate array that implements address and control signal decoding.
- The TL16C452 peripheral.
- The serial connectors J202 and J203.
- The line driver/receiver devices, MAX237 and MAX238.
- The serial clock circuit (for Baud rate synchronization). What is the clock frequency?

By examining the schematic of the TL16C452 more closely, identify the interface of the following pins with the SBC188 system bus: DB0-DB7, A0-A3, /CS1, /IOR, /IOW, /RESET and INT1. You should also identify the interface of the following pins with the MAX 238 line driver/receiver and connector J203: RLSD1, DTR1, SOUT1, CTS1, SIN1, RTS1, DSR1, RI1. Consult pages 4-5 of the TL16C452 Data Sheets in Section 8.2 and read the function descriptions for these pins.

By consulting the IO Port Map section in your Lab Manual, you should verify that the TL16C452 UART associated with J203 is IO-mapped in the SBC188 system as follows.

Port 0x114	IIR, Interrupt Identification Register
Port 0x116	LCR, Line Control Register
Port 0x118	MCR, Modem Control Register
Port 0x11A	LSR, Line Status Register
Port 0x11C	MSR, Modem Status Register
Port 0x11E	SCR, Scratch Register

If DLAB = '0' (bit-7 of the LCR) then

Port 0x110	RBR, Receive Buffer Register (read only)
Port 0x110	THR, Transmit Holding Register (write only)
Port 0x112	IER, Interrupt Enable Register

Otherwise if DLAB = '1' (bit-7 of the LCR) then

Port 0x110	DLL, Divisor Latch (low byte to set Baud rate)
Port 0x112	DLM, Divisor Latch (high byte to set Baud rate)

Read the following sections of the TL16C452 Data Sheets to become familiar with programming the device: page 19 LCR, page 20 LSR, page 23 programmable Baud rate generator, RBR and THR. Also read page 16 and identify the bit assignment Table for all of the serial channel ports.

BACKGROUND : Operation of the Serial Display

The unit displays 1 line of 16-characters at a time. Each character is formed from 14 segments and allows the display of a subset of the ASCII characters including, upper case characters, numerals, punctuation and other common symbols. The serial display unit contains a vacuum fluorescent display and the interface electronics for accepting asynchronous serial characters that adhere to RS-232C. The format is 8-bit characters, 2-stop bits with no parity. The Baud rate is

(jumpered on the unit for) 1200 Baud.

The display unit must also be jumpered for RS-232C operation as shown in Figure 6.3.1.

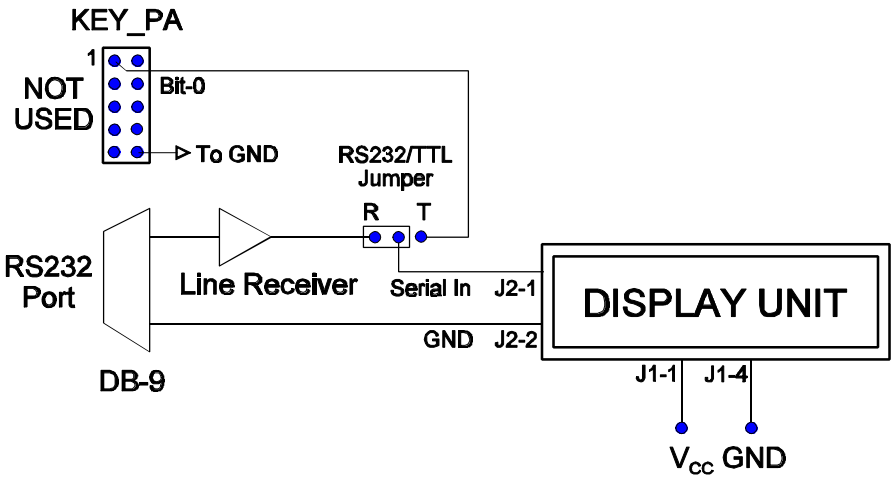


Figure 6.3.1. Interface for the Serial Display

If the serial input is reset to logic '0' (i.e., the driving UART is programmed for transmitting a ***break***) for longer than 4 seconds then the display unit enters a self-test mode where each character from 20H through 5FH is displayed at 1 second intervals.

For proper operation of the display unit, no data should be sent for at least 500 ms after power-up or for 700 microseconds after a software reset. After power-on or software reset the display unit enters a default state with: the cursor in the left-hand position, cursor off, automatic carriage return, display cleared with no character blink.

Special control characters:

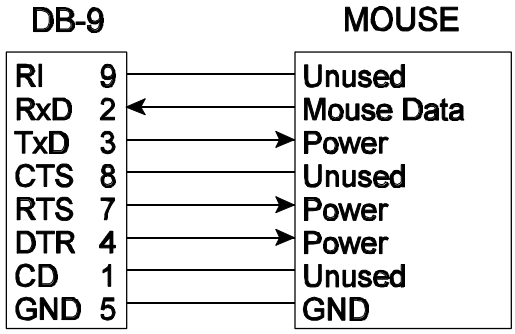
Back space cursor 1-position	08H
Advance cursor 1-position	09H
Line feed	0AH (clears display)
Carriage return	0DH (cursor to position 1)
Automatic carriage return	11H (at end of line)
Over-write right most char.	12H (at end of line)
Horizontal scroll	13H (at end of line)
Start blink field	0BH (following characters blink)
Stop blink field	0CH (following characters don't blink)
Make cursor visible	0FH
Make cursor invisible	0EH
Move cursor (1st byte)	1BH (2nd byte = desired position)

BACKGROUND : Operation of the Serial Mouse

When a (Microsoft 2-button) serial mouse is to interfaced to a RS-232C serial channel the pins

TxD, /DTR and /RTS are used to power the mouse and must be initialized as follows.

TxD = 12V on RS-232C line (the idle state)
/DTR = '0' = 12V on RS-232C line.
/RTS = '0' = 12V on RS-232C line.



/DTR and /RTS may be cleared to '0' by writing 0x03 to the modem control register, MCR, of channel-1 of the TL16C452.

The mouse transmits asynchronous serial characters. The serial format is: 7-bit characters, 1-stop bit, no parity and a Baud rate of 1200. The mouse transmits a 3-character packet whenever its state changes (i.e., whenever the mouse state changes, character 1 is transmitted first then character 2 is transmitted and then character 3 is transmitted). The packet has the format,

Character	**MSB**							**LSB**
First	x	1	LB	RB	y_7	y_6	x_7	x_6
Second	x	0	x_5	x_4	x_3	x_2	x_1	x_0
Third	x	0	y_5	y_4	y_3	y_2	y_1	y_0

NOTE: Bit-6 may be used to synchronize the software for receiving packets. The first character of a packet always begins with bit-6 = '1'. LB denotes the left button ('0' when pressed) and RB denotes the right button ('0' when pressed). $y_7\ y_6\ y_5\ y_4\ y_3\ y_2\ y_1\ y_0$ and $x_7\ x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$ denotes the change in the X and Y directions (in 8-bit 2's complement representation) since the last packet.

BACKGROUND : The Magnetic Card Reader

The magnetic stripe reader transmits a packet of serial data following a swipe of the card across the magnetic read head. The reader may be interfaced to the RS-232C channel on the IO board at the DB-9 connector. The serial data is transmitted asynchronously at 1200 Baud with 7-bit characters, even parity and 1-stop bit. Ensure that the DIP switches on the card reader are set as follows before applying power to the card reader.

SWA							
ON	ON	OFF	ON	OFF	ON	OFF	ON

SWB							
ON	ON	OFF	OFF	ON	ON	OFF	OFF

Each packet transmitted by the card reader begins with the ASCII character ESC (0x1B) followed by the start sentinel, SS, (ASCII '%' for track 1, ASCII ';' for track 2), a (variable) number of card data characters, end sentinel, ES, (ASCII '?') and card status, CS, (ASCII '0' : No card, ASCII '1' : Card in reader), followed by the terminating character, ETX (0x03).

ESC	SS	Card Data: Track 1 Data followed by Track 2 Data	ES	CS	ETX

PREPARATION : Software Development

1. Write a procedure **void DISPLAY_INIT(void)** that performs the following initialization sequence for the TL16C452 Serial Channel 1 whose base address is 110H,

   - Set bit DLAB = 1 of port LCR so the divisor latches can be written for programming the desired Baud rate.
   - Write 0x0060 to the divisor latch i.e., write the 8-most significant bits to port DLM and the 8-least significant bits to port DLL to set the Baud rate = (1.8432 MHz ext. clock ÷ 16) ÷ 96 = 1200 Baud.
   - Write to the LCR port to: disable parity with 2-stop bits and 8-bit characters (with DLAB = 0).

2. Create a properly documented function or procedure **void UART_PUT_CHAR(char X)** that writes the character X to Serial Channel 1. The subroutine must poll the ***transmit holding register empty*** status bit, THRE, (bit-5 of port LSR) until THRE = '1' before writing the data to the transmit holding register, THR,

   ```
   while(THRE == 0) { }
   outportb(THR, X);
   ```

3. Using functions DISPLAY_INIT() and UART_PUT_CHAR(), write a properly documented main program, **CHAR_REPEAT**, that initializes the UART and then continuously sends the same character, say, 'A', over and over again to the UART transmitter.

4. Using UART_PUT_CHAR(), write a procedure **void UART_PUT_STRING(char *PTR_SRC)** that writes a string of characters to the UART transmitter. Assume the starting address of the string is passed as PTR_SRC and the string is terminated with the character, NULL = '\0' = 0x00.

5. Write a procedure **void MOUSE_INIT(void)** that performs the following initialization sequence for the TL16C452 Serial Channel 1 whose base address is 110H,

   - Program the UART for a Baud rate of 1200 Bd as in Part 1.
   - Write to the LCR port to: disable parity with 1-stop bit and 7-bit characters (with DLAB = 0).
   - Set bit-0 = '1' and bit-1 = '1' of MCR to activate DTR* and RTS*.

6. Create a properly documented function or procedure **unsigned char UART_GET_CHAR(void)** that polls the data ready status, DR, (bit-0 of port LSR) until DR = '1', reads the character at the receive buffer register, i.e., port RBR, and returns the character read.

   ```
   while(DR == 0) { }
   ```

return (inportb(RBR));

NOTE: the serial mouse returns a 7-bit character code in ASCII with a parity bit appended as the MSB. If the UART is programmed to receive 8-bit characters (instead of 7-bit) then the MSB of the received character is the transmitted parity bit!

7. Using function UART_GET_CHAR(), write a properly documented function **MOUSE_GET_PACKET(char *PTR)** that reads and returns three consecutive characters (a packet) from the mouse. Assume the three characters are returned as an array whose address is passed as PTR. The first character must be received from the mouse with bit-6 = '1' and the next two characters must each be received with bit-6 = '0' otherwise the function continues to wait for the correct 3-byte sequence to be received.

8. Using functions MOUSE_INIT() and MOUSE_GET_PACKET(), write a program that initializes the UART and then waits for a packet to be sent by the mouse. Display each packet as it is received by main(). You might consider decoding the packet information such that the value of X and Y are written to the hex displays on the IO board and LB and RB are written to the LEDs on the IO board!

9. * Write a procedure **void CARD_INIT(void)** that performs the following initialization sequence for the TL16C452 Serial Channel 1 whose base address is 110H,

   - Program the UART for a Baud rate of 1200 Bd as in Part 1.
   - Write to the LCR port to program : even parity with 1-stop bit and 7-bit characters (with DLAB = 0).

10. * Design a procedure, **void CARD_GET_PACKET(char *BUFF)**, (for interfacing with the magnetic card reader) that waits until ESC is read at the receive buffer register, RBR. (NOTE: you must always poll DR in the LSR before reading RBR) and then reads the packet data, storing each character into the next consecutive location in memory pointed to by BUFF. Assume the first two entries of BUFF contain the ASCII characters CR and LF, and the last character written to BUFF is the string terminating character, NULL = '\0' = 00H.

11 * Using the procedures, PRN_INIT() and PRN_PUT_STRING() from Experiment 6.2, and CARD_INIT() and CARD_GET_PACKET() above, write a program that reads the magnetic card and then prints the card data.

* **ADVANCED SECTIONS**

EXPERIMENT

1. Identify the TL16C452, the serial clock circuit, the MAX238 line driver/receiver and the connector J203 on the SBC188 board.

2. Connect the 10-pin ribbon cable from J-203 on the SBC188 to the 10-pin RS-232C header

on I/O board. Then connect the display unit to the DB-9 connector on the IO board.

3. **Hardware Debugging:**

   Using the Debug Monitor I/O command, initialize Serial Channel 1 of the TL16C452 by writing the control words in the same sequence as programmed in the procedure, DISPLAY_INIT().

   Now use the Debug Monitor I/O command to read the UART line status register, LSR, and examine the state of the DR and THRE status bits. Test the hardware interface using the Debug Monitor I/O command by writing an ASCII character to the UART transmitter and watching the terminal display unit.

4. Load and debug the program, **CHAR_REPEAT**. Connect a scope to the TxD line of the RS-232C channel and carefully examine and verify the details of the asynchronous communication, i.e., voltage and logic levels, the proper sequence for start bit, data bits, parity bit, stop bits and Baud rate.

5. Using **UART_PUT_STRING()**, write a main program that displays your name at the display unit. How would you modify the program to scroll your name from left to right over and over again?

6. Now connect mouse to the DB-9 connector on the IO board. You may use the Debug Monitor I/O command to read the UART line status register, LSR, examine the state of the DR status bits and read the data at the ***receive buffer register***, RBR.

7. Load and debug the mouse packet reading program of Part 8 of the Preparation.

8. * Connect the serial card reader to the DB-9 connector on the IO board. Load and debug the card packet reading program of Parts 9 to 11 of the Preparation.

SUMMARY

1. Draw a diagram tat shows the connection between the TL16C452 device and system bus.

2. Using a timing diagram sketch the asynchronous serial protocol when ‘A’ is transmitted. What happens to the parity bit if odd, even or stick parity is enabled?

3. Why is handshaking with the receiver and transmitter necessary?

4. For the Baud rates used in this experiment, what are the corresponding (maximum) character transfer rates in characters per second?

5. What happens (or what should happen) in the procedure UART_GET_CHAR() if a receive error, such as overrun, parity or framing occurs during reception of serial data?

EXTENSIONS : Implementing Interrupt Driven IO with Serial Devices

The following describes how you may modify the polled handshaking design so that the Serial Channel 1 of the TL16C452 is interrupt driven. For example the TL16C452 allows the transmitter and receiver to interrupt the CPU whenever the transmit holding register is empty or when the receiver is ready with a character. How would you modify UART_GET_STRING() so as to implement interrupt driven handshaking? What are the advantages and disadvantages of the interrupt driven interface over the polled handshaking used in this experiment?

Refer to Figure 6.5.1 for the configuration of the interrupt controllers on the SBC188. To implement an interrupt driven interface, first choose an interrupt type number, (say 0xFA for example) for the Serial Channel 1 interrupt that is connected to IR2 of the slave 8259A interrupt controller shown in Figure 6.5.1. Then initialize the interrupt vector table for the interrupt handler associated with Serial Channel 1.

To enable interrupt generation (by either serial channel of the TL16C452) bit-3 of port MCR must be set, (in addition to bit-0 and bit-1). Certain bits within the port IER also need to be set to enable specific serial interrupts. For example to enable RECEIVER interrupts: set bit-0 of the IER port. This causes the TL16C452 to generate an interrupt request at the INT1 output pin whenever a character is received by the UART receiver and that character is now ready to be read by the CPU. Refer to pages 16-17 the TL16C452 Data Sheets.

The TL16C452 serial interrupt drives IR2 of a slave 8259A interrupt controller. The slave interrupt controller drives a master 8259A at IR2 (much like the interrupt generation in a PC computer). And the master 8259A interrupt controller drives the external interrupt pin INT0 of the 80C188XL microprocessor as shown in Fig 6.5.1. Before interrupts can be recognized by the microprocessor the interrupt controllers all need to be initialized as follows:

Slave 8259A in the 82C206: (pages 22-29 in the 82C206 Data Sheets)

- Write byte 0x19 to ICW1 (address 0x00A0 in the 82C206) to program level triggered cascade operation.
- Write the desired 8-bit interrupt type (base number) to ICW2 (address 0x00A1 in the 82C206) to program the base type number for interrupt requests at the slave interrupt controller, say 0xF8.
- Write byte 0x02 to ICW3 (address 0x00A1 in the 82C206) indicating the slave controller address.
- Write byte 0x01 to ICW4 (address 0x00A1 in the 82C206) to program single interrupt nesting and normal end of interrupts.

Master 8259A in the 82C206: (pages 22-29 in the 82C206 Data Sheets)

- Write byte 0x19 to ICW1 (address 0x0020) to program level triggered cascade operation.
- Write the desired 8-bit interrupt type (base number) to ICW2 (address 0x0021) to program the base type number for interrupt requests at the master interrupt controller, say 0xF0.
- Write byte 0x04 to ICW3 (address 0x0021) indicating that a slave interrupt controller connected to IR2 of the master interrupt controller.

- Write byte 0x01 to ICW4 (address 0x0021) to program single interrupt nesting and normal end of interrupts.

80C188XL Interrupt Controller: (page 8-15 in the 80C188XL Users Manual)

- Write word 0x0030 to I0CON (address 0xFF38) to program hi-level trigger, cascade mode, not fully nested with priority 000, (highest).

Unmask the Master 8259A IR2 and Slave 8259A IR2:

- Write byte 0xFB to OCW1 of the Master (address 0x0021) to unmask IR2 in master.
- Write byte 0xFB to OCW1 of the Slave (address 0x00A1) to unmask IR2, i.e., the Serial Channel 1 interrupt requests at the slave.

Within the interrupt handler (i.e., interrupt service routine) there is a need to send an ***end of interrupt command*** to each of the interrupt controllers as follows (in order for further interrupts to be processed).

- Write byte 0x62 to OCW2 in the Slave (address 0x00A0) to signify that the interrupt service routine for the slave request, IR2, has finished execution.
- Write byte 0x62 to OCW2 in the Master (address 0x00A0) to signify that the interrupt service routine for the master request, IR2, has finished execution.
- Write word 0x000C to the 16-bit port EOI in the 80C188XL microprocessor (address 0xFF22) to signify that the interrupt service routine for hardware interrupt INT0 has finished execution.