An Evaluation of the 8051 Microcontroller

Spring 1983

George C. Schafner

University of Central Florida

Find similar works at: http://stars.library.ucf.edu/rtd

University of Central Florida Libraries http://library.ucf.edu

Part of the Engineering Commons

STARS Citation

http://stars.library.ucf.edu/rtd/717

This Masters Thesis (Open Access) is brought to you for free and open access by STARS. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of STARS. For more information, please contact lea.dotson@ucf.edu.
ABSTRACT

With the increasing availability and use of 16-bit microprocessors, the 16-bit data bus is becoming more prevalent. However, many peripheral devices such as printers and tape/disk drives still require an 8-bit data bus for their interface.

This paper will explain how an Intel 8051 microcontroller may be used to interface a 16-bit data bus to a peripheral requiring an 8-bit data bus. A FIFO is used to buffer data from a 16-bit processor so that efficient use of processing time is maintained. The 8051 is used to control the peripheral and data transfer.
AN EVALUATION
OF
THE 8051 MICROCONTROLLER

BY
GEORGE C. SCHAFNER
B. S. E., Auburn University, 1978

RESEARCH REPORT
Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
in the Graduate Studies Program of the College of Engineering
University of Central Florida
Orlando, Florida

Spring Semester
1983
ACKNOWLEDGEMENTS

The author wishes to thank Burroughs Corporation's Orlando Special Printers Group for providing the resources necessary to perform the tests described within this paper. Thanks also to Dr. Herbert C. Towle for his guidance and direction in conducting this research. And special thanks to my wife, Nancy, for her support, encouragement and typing skills.
TABLE OF CONTENTS

I. INTRODUCTION ............................................. 1
II. 8051 DESCRIPTION AND 8048 COMPARISON .............. 2
III. 8051 APPLICATION POSSIBILITIES ...................... 7
IV. DESCRIPTION OF SYSTEM ................................. 12
    Test System ............................................. 33
    Performance ............................................. 35
V. CONCLUSION ............................................... 41

Appendix

A. TEST CIRCUIT SCHEMATIC ................................. 42
B. TEST PROGRAM FLOWCHARTS AND PROGRAM LISTING .... 45
C. FIFO SPECIFICATION ..................................... 65
D. CENTRONICS-COMPATIBLE INTERFACE SPECIFICATION . 74

LIST OF REFERENCES ........................................ 85
I. INTRODUCTION

This research paper is an evaluation of the Intel 8051 microcontroller. This evaluation is accomplished by analyzing the 8051 from three different aspects. First, its features are studied and an 8048 microcontroller comparison is included. Then some current and possible applications are discussed. Thirdly, the design of a specific application is described step by step and then tested for performance.

In the demonstration, a first-in first-out (FIFO) storage device along with 16-bit to 8-bit data conversion is used to complement the 8051's performance and capabilities. A Centronics Interface compatible printer (Data Products M-200) is controlled by the 8051 and the data to be printed originate at a 16-bit bus. The FIFO, Monolithic Memories' 67401, serves as a buffer between the 16-bit bus and the printer/controller.

An objective evaluation of the 8051's capabilities and performance is included in the conclusion. Detailed test circuit schematics, flow charts and program listings are included in this paper. And finally, specifications for the 67401 FIFO and the M-200 Centronics compatible interface are included for convenient reference.
II. 8051 DESCRIPTION AND 8048 COMPARISON

The 8051 is an 8-bit microcontroller with features that allow it to be used in a wide variety of applications, some of which will be discussed in the next chapter. The 8051 contains 4K bytes of program memory (ROM) and 128 bytes of data memory (RAM). There are four 8-bit input/output ports where each bit may be used as either an input or an output. Two 16-bit programmable timer/event counters are available and provide two of five possible sources of interrupt. External events may be used to interrupt the 8051 via the INT0 and INT1 pins. A full-duplex serial I/O port generates an interrupt at the completion of transmission or reception of one serial frame of bits. The 8051 may be considered to be a Boolean processor with its set of bit-orientated instructions (set, clear, complement, and, or, move and jump-on bit). When a 12 MHz crystal is used, most instructions can execute in 1 microsec. A block diagram of the 8051 is shown in figure 1 (1, 2, 3).

The program and data memory may be expanded externally to provide up to 64K-bytes for each.

Since Intel's 8048 is the 8051 predecessor it is beneficial to make a comparison of these two controllers. Table 1 shows a comparison of some features which would be considered for various applications (1, 2).
Figure 1. 8051 block diagram.
<table>
<thead>
<tr>
<th>FEATURE</th>
<th>8048</th>
<th>8051</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O PORTS</td>
<td>27</td>
<td>32</td>
</tr>
<tr>
<td>PROGRAM MEMORY</td>
<td>1 K-Bytes</td>
<td>4 K-Bytes</td>
</tr>
<tr>
<td>DATA MEMORY</td>
<td>64 Bytes</td>
<td>128 Bytes</td>
</tr>
<tr>
<td>TIMER/EVENT COUNTER</td>
<td>One 8-Bit</td>
<td>Two 16-Bit</td>
</tr>
<tr>
<td>INSTRUCTION CYCLE (minimum)</td>
<td>2.5 µsec</td>
<td>1.0 µsec</td>
</tr>
<tr>
<td>INTERRUPTS</td>
<td>Two Sources</td>
<td>Five Sources</td>
</tr>
<tr>
<td></td>
<td>Single Level Priority</td>
<td>Two Level Priority</td>
</tr>
<tr>
<td>REGISTERS</td>
<td>Two 8-Register Banks</td>
<td>Four 8-Register Banks</td>
</tr>
<tr>
<td>FULL-DUPLEX SERIAL CHANNEL</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>MULTIPLY &amp; DIVIDE INSTRUCTIONS</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>EXTERNAL MEMORY EXPANSION</td>
<td>4.3 K-Bytes</td>
<td>128 K-Bytes</td>
</tr>
</tbody>
</table>
The pulsing of a particular signal via software is a common practice in controller applications. It is therefore worthwhile to make a comparative evaluation of the software required for this practice.

If it is desired that bit 1 of port 1 be pulsed, the following routine would be required if the 8048 were used:

```
ORL P1,#01H
ANL P1,#0FEH
```

Since these are 2-cycle instructions, there would be a 10.0 microsecond execution time.

If the 8051 were used, the following routine would be required:

```
SETB P1.0
CLR P1.0
```

This would result in a 2.0 microsecond execution time since these are 1-cycle instructions. Therefore, a pulse could be generated five times faster by the 8051 than by the 8048.

It should be noted that the bit-orientated instructions greatly enhance the 8051 instruction set.

Another common practice in controller applications is the output of data to a port. The data must go through the accumulator when using the 8048, whereas it can be output
directly to the port when using the 8051. This comparison is shown below.

<table>
<thead>
<tr>
<th>8048</th>
<th>8051</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV A,#OACH</td>
<td>MOV Pl,#OACH</td>
</tr>
<tr>
<td>OUTL Pl,A</td>
<td></td>
</tr>
<tr>
<td>4 cycles = 10.0 microsec.</td>
<td>2 cycles = 2.0 microsec.</td>
</tr>
</tbody>
</table>

The 8051 is a more powerful controller than the 8048, but each application must be evaluated to determine if the additional 8051 features are beneficial. Various application possibilities will be discussed in the next chapter.
III. 8051 APPLICATION POSSIBILITIES

The possible applications for an 8051 are virtually limitless. A few of the more common applications will be discussed in this chapter.

Since the 8051 has all the capabilities of the 8048, it can be used as a replacement for the 8048. This replacement may be desired when the application code in the 8048 expands beyond 1K bytes or when faster code execution is needed. One common use of the 8048 is a control processor for a bar-code scanner (4, 5, 6). Acting as a slave to a host computer, the data can be sent in an 8-bit parallel format or in a serial format such as RS-232-C. For a parallel format, the 8051 could serve as a replacement when an increased baud rate is desired. In the case where serial format is desirable, the 8051 could replace the 8048 by utilizing its serial port (RXD, TXD) capabilities shown in figure 2 (3). A write to the Special Function Register, SBUF, loads the transmit register while a read accesses a separate receive register. Another Special Function Register, SCON, is used to establish a desired operating mode: (a) Mode 0 - 8 data bits transmitted/received at a baud rate of 1/12 the oscillator frequency (max. = 1MHz). (b) Mode 1-10 bits transmitted/received
Figure 2. 8051 serial port (modes 1, 2, and 3)
which include a start bit, 8 data bits, and a stop bit. The baud rate is variable from 122 to 31,250 bits per second.

(c) Mode 2-11 bits transmitted/received which include a start bit, 8 data bits, a programmable 9th data bit, and a stop bit. The baud rate is set for 1/32 or 1/64 of the oscillator frequency. (d) Mode 3-11 bits transmitted/received same as for Mode 2. The baud rate is variable, same as for Mode 1 (1).

Keyboard scanning and display control can be accomplished through the use of an 8051 (2). To more fully utilize its potential the 8051 could be delegated additional control and/or communication responsibilities. A 24-hour banking station is an example of this application. The 8051 scans the keypad and updates the display based upon the customer's entry. Meanwhile, the 8051 and a host computer relay the pertinent account information. A printout of the transaction could also be generated through the 8051's control. A possible configuration for a 24-hour bank station control system is shown in figure 3. A detailed design of this particular application is not the purpose of figure 3. Instead, the flexibility of the 8051 is demonstrated by permitting a wide variety of peripherals to be controlled on a time-shared basis. When the 8051 I/O ports are exhausted, a port expander (8243) may be attached. The 8243 uses six I/O pins from the 8051 to generate four 4-bit ports (1). Test pins, T0 and T1,
Figure 3. Block diagram of 24-hour bank station control system
can be used to monitor status of the peripherals and interrupts, INT0 and INT1, can be used for priority conditions. RXD and TXD can be connected to a peripheral which uses a serial interface.

Other applications for the 8051 include tape and disk transport controllers. Tape and disk positions can be monitored while data and transport are controlled. Direction, motion, speed, and read/write select are examples of the controllers responsibilities.

A brief list of current in use applications for the 8048 and 8051 controllers includes microwave oven control, automotive carburetor feedback control, gas pump control, automotive dashboard control, washing machine control, traffic light control, and heating control (1, 2).

Most, if not all of the applications discussed and mentioned in this chapter make use of 8-bit and/or serial data communication. The increasing use of 16-bit processors and the 16-bit data bus should be considered in conjunction with the various 8-bit orientated peripheral control applications (7, 8). This will be accomplished through a specific design in the next chapter.
IV. DESCRIPTION OF SYSTEM

A detailed description of an 8051 application is presented in this chapter. A FIFO buffering technique (9, 10) is used to complement the features of the 8051 while being used as a printer controller. The FIFO is used to link a 16-bit data bus to the controller which transfers 8-bit data to the printer. The widely used Centronics Interface links the controller to the printer (7). A block diagram of the system is shown in figure 4. The principal objective of this system is to maximize the data transfer rate to the printer while minimizing the 16-bit bus access time. Monolithic Memories' 67401 FIFO can be used to provide a 10 MHz data rate from the 16-bit bus (11). It is therefore possible for 64 words, 128 bytes, to be stored in the FIFO in 6.4 microseconds. The FIFO not only allows high speed data reception and storage but it also serves as a 16 to 8 bit data converter. The 67401 is a 64X4 FIFO which means that four are required to achieve the desired word width. At the 16-bit data bus input, the FIFO functions as a 64X16 device when the "input ready" (IR) signals are anded together and the "shift-in" (SI) control is a common signal for each of the four FIFO segments. At the output, the FIFO functions as two 64X8 devices having separate
Figure 4. System block diagram
"output ready" (OR) and "shift-out" (SO) lines that are monitored and controlled by the 8051. A detailed specification, including timing diagrams, for the 67401 can be found in Appendix C. During optimum operation conditions, the 16-bit data bus controller monitors the combined "output ready" signals and shifts in a 64 word burst of data when the FIFO is empty. The order of desired print is as follows:

<table>
<thead>
<tr>
<th>UPPER BYTE</th>
<th>LOWER BYTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>64th Word</td>
<td>127th</td>
</tr>
<tr>
<td></td>
<td>Last Byte</td>
</tr>
<tr>
<td></td>
<td>To Be</td>
</tr>
<tr>
<td></td>
<td>Printed</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>2nd Word</td>
<td>5th</td>
</tr>
<tr>
<td></td>
<td>4th</td>
</tr>
<tr>
<td></td>
<td>3rd</td>
</tr>
<tr>
<td>1st Word</td>
<td>1st Byte</td>
</tr>
<tr>
<td></td>
<td>To Be</td>
</tr>
<tr>
<td></td>
<td>Printed</td>
</tr>
<tr>
<td></td>
<td>2nd</td>
</tr>
</tbody>
</table>

The 8051 alternates shifting out the lower byte from the first 64x8 FIFO and then the upper byte from the second 64x8 FIFO. Separately enabled drivers, LS244s (12) are selected by the 8051 as to which byte is sent to the printer or to the 8051. Since the output data of the FIFO remains valid as long as "shift-out" is active, this same signal can be used for the driver enables.

Port assignment is an important consideration when using the 8051 in a particular design. The signals with the more
critical timing constraints are considered first. The timing diagram in figure 5 shows the more critical signals during transfer for the Centronics-Compatible Interface (13). The "data strobe" is generated by the 8051 and cannot be greater than 1.0 microsecond which means that setting and clearing a pin is not practical. Also to maximize the data transfer rate a 0.5 microsecond pulse is desired. By using the write pulse at pin 6 of port 3 this 0.5 microsecond "data strobe" is achieved. The write pulse is generated during a data memory write cycle as shown in figure 6 (3). The minimum WR pulse width = (6) (P) - 100 nanoseconds, where "P" is the oscillator period. A 10 MHz oscillator is therefore needed for a 0.5 microsecond WR pulse. Since port 0 is a multiplexed address and data port, it cannot serve as the data port to the printer due to insufficient setup time. Port 2 is used as an address port when an external data memory move command occurs for a 16-bit address. For this particular application, only 8-bit external data memory move commands are used. Hence, either port 1 or port 2 may be used as the data port. Having selected port 1 for the data port, port 2 is used to monitor and control signals with minor timing restrictions.

The "acknowledge" pulse generated by the printer must be captured by the 8051 to maintain proper handshaking. Since
Figure 5. Data transfer timing
(Centronics-compatible interface)

T1 > 0.0 \mu \text{Sec}
T2 = 1.0 \mu \text{Sec.min.}
T3 = 0.0 \mu \text{Sec.min.}
= 0.075 \mu \text{Sec.max.}
T4 = 0.5 \mu \text{Sec.min.}
= 1.0 \mu \text{Sec.max.}
T5 = 1.0 \mu \text{Sec.min.}
T6 > 5.8 \mu \text{Sec.}
T7 = 3.0 \mu \text{Sec.min.}
= 5.0 \mu \text{Sec.max.}
Figure 6. 8051 data memory write cycle
this pulse has a minimum duration of 3 microseconds, a pin can be sampled using a bit instruction such as "JB P3.4, $". This is a two cycle instruction, 2.4 microseconds, that allows program execution to continue when pin 4 of port 3 goes low. "Acknowledge" should not be used to generate an external interrupt since the service routine would cause needless delay.

If a "Fault" occurs during a data transfer then the 8051 should stop the transfer. This condition takes place when the printer is out of paper, the shuttle is not moving, or when the printer is deselected. The "Fault" signal is assigned to the INTI external interrupt location at pin 3 of port 3.

The remaining Centronics Interface signals have less critical timing restrictions so they are assigned to the following arbitrary locations:

Pin 0 of Port 3 - "Input Prime" (Generated by the 8051 to clear the printer buffer and initialize the interface logic)

Pin 1 of Port 3 - "Select" (Generated by the printer when it is on line and ready for data transfer)

Pin 0 of Port 2 - "Busy" (Generated by the printer when it is unable to receive print or format data)

Pin 1 of Port 2 - "PE" (Generated by the printer when it is out-of-paper)
If it is desired that one of the internal counters of the 8051 be used to monitor the amount of data transferred then the "acknowledge" signal can be attached to T0 or T1 of port 3. For a detailed description of the Centronics-Compatible Interface, see Appendix D.

The FIFO output control and the 8-bit data path control is accomplished with port 2 locations. Port 0 has not been delegated any control responsibilities primarily due to its data/address multiplexing during the external memory move commands. Therefore, port 0 is available if the capabilities of this control system need to be expanded. For example, if it is desired that the 8051 read data from remote memory then the RD signal, pin 7 of port 3, can be used with port 0 as shown in figure 7 (3). Data bytes can be read from 256 separately programable memory locations.

Both the 8051 and the FIFO may be externally reset if an error occurs or if complete initialization is desired. The FIFO "master reset" clears the FIFO while the 8051 "reset" initializes the program counter to zero and sets all I/O port locations to 1.

A logical control program sequence for the FIFO/Centronics Interface is shown by the flowchart in figure 8. However, when this sequence is used, needless delays occur after "acknowledge" pulses are received. Therefore the
Figure 7. 8051 data memory read cycle (for MOVX A, @Ri command)
Figure 8. Flowchart of logical control sequence
sequence is modified for reduced delay as shown by the flowchart in figure 9. The SELGEN, "Select" generation, subroutine selects the printer by sending a Hex 11 over the data lines. Once "select" is received, SELGEN enables the "fault" interrupt which occurs whenever "select" goes inactive. A flowchart for SELGEN subroutine is shown in figure 10. A listing of the code generated from figures 9 and 10 follows those flowcharts.

If the data to be printed are hex data (i.e. 0 - F) then the storage capacity of the FIFO can be doubled. Nibbles can be converted into ASCII equivalents for the hex values to be printed. Table 2 shows the required conversion and a conversion routine is shown by the flowchart in figure 11. The corresponding code follows the flowchart.
INITIALIZATION

PULSE "INPUT PRIME" TO RESET PRINTER

CALL SELGEN ("SELECT" GENERATION)

FIFO OUTPUT READY?

SHIFT OUT LOWER BYTE

GENERATE "DATA STROBE" PULSE USING MOVX INSTRUCTION

"ACKNOWLEDGE" RECEIVED?

SHIFT OUT UPPER BYTE

"ACKNOWLEDGE" RECEIVED?

FIFO OUTPUT READY?

GENERATE "DATA STROBE" PULSE USING MOVX INSTRUCTION

"ACKNOWLEDGE" RECEIVED?

Figure 9. Flowchart of reduced delay sequence
Figure 10. Flowchart for SELGEN ("Select" generation) subroutine
LOC OBJ LINE SOURCE

1 //**********************************************************************************
2 //**********************************************************************************
3 //**********************************************************************************
4 //******** TITLE: 8051 FIFO/CENTRONICS CONTROLLER
5 //**********************************************************************************
6 //******** DATE: FEB. 1, 1983
7 //**********************************************************************************
8 //******** PROJECT: EEL 6918 (GRADUATE RESEARCH)
9 //**********************************************************************************
10 //******** AUTHOR: G. C. SCHAFTER
11 //**********************************************************************************
12 //**********************************************************************************
13 //**********************************************************************************
14 ;
15 ;
16 ; INPUT PIN DECLARATIONS
17 ;
18 00A0 BUSY BIT P2.0 ;BUSY FROM PRINTER (LOW TRUE)
19 00A1 PB BIT P2.1 ;PAPER EMPTY (LOW TRUE)
20 00A2 ORDY BIT P2.2 ;FIFO OUTPUT READY (LOW TRUE)
21 00A3 UBOK BIT P2.3 ;FIFO UPPER BYTE OUTPUT READY (LOW TRUE)
22 00B1 SELECT BIT P3.1 ;SELECT FROM PRINTER (LOW TRUE)
23 00B2 EX BIT P3.2 ;EXECUTE REQUEST (LOW TRUE)
24 00B3 FAULT BIT P3.3 ;FAULT FROM PRINTER (LOW TRUE)
25 00B4 ACKNL6 BIT P3.4 ;ACKNOWLEDGE FROM PRINTER (HIGH TRUE)
26 ;
27 ;
28 ; OUTPUT PIN DECLARATIONS
29 ;
30 00A4 LBSO BIT P2.4 ;LOWER BYTE SHIFT OUT (LOW TRUE)
31 00A5 UBSO BIT P2.5 ;UPPER BYTE SHIFT OUT (LOW TRUE)
32 00B0 IMPRI BIT P3.0 ;INPUT PRIME TO PRINTER (LOW TRUE)
33 00B6 STKB BIT P3.6 ;DATA STROBE TO PRINTER (LOW TRUE)
34 ;
35 ;
36 +1 $EJECT
**INTERRUPT SERVICE ROUTINES**

```assembly
ORG OOH
AJMP INIT ;GO TO INIT AT RESET

ORG 03H ;EXECUTE INTERRUPT ROUTINE
CLR EA ;DISABLE INTERRUPTS
CLR EX0 ;DISABLE EXTERNAL INTERRUPT 0
AJMP CONT ;CONTINUE

ORG 0BH ;40mSEC TIMER ROUTINE
CLR EA ;DISABLE INTERRUPTS
CLR ET0 ;DISABLE TIMER 0 INTERRUPT
CLR TR0 ;DISABLE TIMER 0
AJMP EXEC ;WAIT FOR EXECUTE REQUEST

ORG 13H ;"FAULT" INTERRUPT ROUTINE
CLR EA ;DISABLE INTERRUPTS
AJMP INTRET ;GO TO INTRET

ORG 18H ;75mSEC TIMER ROUTINE
CLR EA ;DISABLE INTERRUPTS
CLR ET1 ;DISABLE TIMER 1 INTERRUPT
CLR TR1 ;DISABLE TIMER 1
AJMP CONT ;CONTINUE

+1 $EJECT
```
**INITIALIZATION:** This routine sets up I/O

Ports, awaits execution request, and resets the printer.

```
0043 74FF
0045 F590
0047 F5A0
0049 F5B0
004B 75B911
004E C2B0
0050 75BAC9
0053 75BC7D
0056 D29A
0058 D2AF
005A D2BC
005C 80FE
005E D2B0
0060 75B107
0063 90006B
0066 C082
0068 C083
006A 32
006B D2AB
006D D2AF
006F 80FE
0071 75B107
0074 90007C
0077 C082
0079 C083
007B 32
007C 11C9
007E D2AF
0080 01AB
0084 ORG 43H ;SET UP I/O PORTS
0085 MOV A, #0FFH
0086 MOV P1A
0087 MOV P2A
0088 MOV P3A
0089 MOV TMOD, #11H
008A CLR INPRT
008B MOV TLO, #0C9H
008C MOV TH0, #07DH
008D SETB ET0
008E SETB EA
008F SETB TR0
0090 JMP $ ;WAIT FOR TIMOUT
0091 MOV SP, #07H
0092 POP DPL
0093 POP DPH
0094 RETI
0095 ACALL SELGEN ;GENERATE "SELECT"
0096 AJMP CKFIFO ;START TRANSFER
0097 11C9 $EJECT
```
124 ; CHECK FIFO STATUS FOR "FAULT" INTERRUPT
125
126 INTRET: POP DPH  ; REMOVE ADDRESS WHERE INTERRUPT OCCURRED
127 POP DPL  ; LOWER BYTE OF ADDRESS IN DPL
128 MOV R4,DPL  ; SAVE LOWER BYTE IN R4
129 MOV DPTR,#RET3  ; SET UP NEW INTERRUPT RETURN
130 PUSH DPL
131 PUSH DPH
132 RETI
133 RET3: MOV DPTR,#NEXT  ; FIND WHERE INTERRUPT OCCURRED
134 MOV A,DPL
135 CLR C
136 SUBB A,R4  ; COMPARE "NEXT" ADD. TO INTERRUPT ADD.
137 JNC CKFIFO  ; JUMP IF INTERRUPT ADD. =< "NEXT"
138 MOV DPTR,#CHECK
139 MOV A,DPL
140 CLR C
141 SUBB A,R4  ; COMPARE "CHECK" ADD. TO INTERRUPT ADD.
142 JC CKFIFO  ; JUMP IF INTERRUPT ADD. > "CHECK"
143 CLR UB50  ; SHIFT OUT UPPER BYTE
144 SETB EA  ; ENABLE INTERRUPTS
145 AJMP UPPER  ; SEND UPPER BYTE
146 ;
147 ;
148 +1 $EJECT
TRANSFER: This section controls the transfer from the FIFO to the printer.

;=============================================

; CKFIFO: JB ORDY,$  WAIT FOR FIFO READY
CLR LBSO  $SHIFT-OUT LOWER BYTE
MOVX @RO,A $GENERATE "DATA STROBE" (NR PULSE)

NEXT: SETB LBSO $DISABLE LOWER BYTE
CLR UBSO  $SHIFT-OUT UPPER BYTE
JNB ACKNLG,$ $WAIT FOR "ACKNOWLEDGE"

UPPER: MOVX @RO,A $"DATA STROBE" FOR UPPER BYTE
CHECK: SETB UBSO $DISABLE UPPER BYTE
JB ORDY,CKFIFO $IF FIFO NOT READY GO TO CKFIFO
CLR LBSO  $SHIFT-OUT LOWER BYTE
JNB ACKNLG,$ $WAIT FOR "ACKNOWLEDGE"

A JMP NEXT $SEND NEXT BYTE

;=============================================

; SELGEN: This subroutine "selects" the printer
; AND THEN ENABLES THE "FAULT" INTERRUPT.

;=============================================

; SELGEN: MOV P1,#11H $OUTPUT SELECT REQUEST
MOVX @RO,A $GENERATE "DATA STROBE"
MOV TL1,OFFH $LOAD TIMER 1 FOR 75mSEC
MOV TH1,0BH $;

SETB ET1 $ENABLE TIM1 INTERRUPT
SETB EA $ENABLE INTERRUPTS
SETB TR1 $START TIMER 1

JB SELECT,$ $WAIT FOR "SELECT"
CLR EA $DISABLE INTERRUPTS
CLR ET1 $DISABLE TIMER
CLR TR1 $STOP TIMER

MOV P1,#0FFH $SET PORT 1 AS INPUT

SETB EX1 $ENABLE INT1 INTERRUPT ("FAULT")

RET

;=============================================

ASSEMBLY COMPLETE, NO ERRORS FOUND
<table>
<thead>
<tr>
<th>NIBBLE FROM FIFO</th>
<th>HEX EQUIVALENT</th>
<th>ASCII EQUIVALENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
<td>00110000 (30H)</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>00110001 (31H)</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
<td>00110010 (32H)</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>00110011 (33H)</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
<td>00110100 (34H)</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
<td>00110101 (35H)</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
<td>00110110 (36H)</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
<td>00110111 (37H)</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
<td>00111000 (38H)</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
<td>00111001 (39H)</td>
</tr>
<tr>
<td>1010</td>
<td>A</td>
<td>01000001 (41H)</td>
</tr>
<tr>
<td>1011</td>
<td>B</td>
<td>01000010 (42H)</td>
</tr>
<tr>
<td>1100</td>
<td>C</td>
<td>01000011 (43H)</td>
</tr>
<tr>
<td>1101</td>
<td>D</td>
<td>01000100 (44H)</td>
</tr>
<tr>
<td>1110</td>
<td>E</td>
<td>01000101 (45H)</td>
</tr>
<tr>
<td>1111</td>
<td>F</td>
<td>01000110 (46H)</td>
</tr>
</tbody>
</table>
Figure 11. Flowchart of nibble conversion
;******************************************************************************
;********** NIBBLE SAVE: This subroutine **********
;********** receives bytes from the FIFO, converts the **********
;********** nibbles to Hex/ASCII equivalent, and saves **********
;********** the corresponding byte in internal RAM **********
;********** (locations 30Hex to 79Hex). **********
;********** EXIT VALUES: R1 = Address of last data in **********
;********** in internal RAM to be **********
;********** transferred. **********
;******************************************************************************

NIB: MOV R7,#01H ; Upper or lower byte status flag
MOV R1,#30H ; Initialize address location
NXTNIB: JB ORDY,DONE ; Jump if FIFO empty
DEC R7 ; STATUS=0
CLR LB50 ; Shift out lower byte
MOV R4,P1 ; Save lower byte in R4
SETB LB50

CONV: INC R7 ; Status incremented

CONV LOWER NIBBLE

CONV:

MOV A,R4
ANL A,#0FH
ADD A,#90H
DA A
ADDC A,#40H
DA A
MOV @R1,A
INC R1

CONV UPPER NIBBLE

INC R1
DJNZ R7,ADDCK ; Jump if upper byte completed
CLR UB50 ; Shift out upper byte
MOV R4,P1 ; Save upper byte in R4
SETB UB50 ; Status=1
AJMP CONV ; Convert upper nibble

ADDCK: CJNE R1,#80H,NXTNIB ; Jump if address < 80Hex
DONE: RET
**Test System**

To test the proposed system, a circuit is connected to a M - 200 Data Products matrix printer (13) and an Intel In-Circuit Emulator (ICE51) is used to control and monitor program execution (14, 15, 16, 17). A detailed schematic of the test circuit can be found in Appendix A. Flowcharts and program listing for the test code can be found in Appendix B.

Hardware switches are used to vary the 16-bit data to be shifted into the FIFO. The FIFO reset and shift-in control is accomplished using momentary push button switches. Likewise, the 8051 is reset with a momentary push button switch. The 8051 resets and initializes the printer and then awaits an execution request thru an interrupt (INTO) generated by another push button switch. While the 8051 waits for an execution request, data is manually shifted into the FIFO. The first word shifted into the FIFO should contain a coded command in the lower byte and desired number of repetitions for transfer in the upper byte. The available commands are as follows:
<table>
<thead>
<tr>
<th>UPPER BYTE</th>
<th>LOWER BYTE</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>00H</td>
<td>01H</td>
<td>Direct byte transfer from FIFO to printer</td>
</tr>
<tr>
<td>01H-FFH</td>
<td>01H</td>
<td>Save the next 80 FIFO bytes in RAM. Then transfer desired repetitions.</td>
</tr>
<tr>
<td>00H-FFH</td>
<td>02H</td>
<td>Convert the next 80 nibbles (40 bytes) into their Hex/ASCII equivalents. Store the conversion in RAM. Then transfer desired repetitions.</td>
</tr>
<tr>
<td>00H-FFH</td>
<td>03H</td>
<td>Generate a test pattern (59 characters, &quot;SP&quot; - &quot;Z&quot;). Save in RAM. Transfer to printer desired repetitions.</td>
</tr>
</tbody>
</table>

One transfer occurs when the upper byte contains 00H. For an undefined command (00H, 04H - FFH) the data in the FIFO is transferred directly to the printer. Once command execution is completed, the 8051 awaits another execution request. The "fault" interrupt (INT1) is only enabled during data transfer and the "execute" interrupt (INT0) is only enabled when the printer has been reset and no command execution is in progress. A total of 20,480 characters may be printed when the repetition number is set at FFH. Intelligible words and sentences can be constructed using the 01H command. For the test system, the maximum data transfer rate to the printer occurs when there
are 0 repetitions of the 01H command. This is probably the more practical application of the proposed system.

**Performance**

Maximum specified transfer rate of data to the M - 200 printer is 75 Kbytes/sec. (13). This transfer rate is achieved by the test circuitry when data are transferred directly from the FIFO to the printer. Photo 1 shows a logic analyzer trace of such a transfer. Once the first byte has been set up and transferred, then the 75 Kbytes/sec. rate is calculated as follows:

\[
\text{Transfer Rate} = \frac{12 \text{ bytes}}{(1\mu\text{sec}/\text{clock})(20\text{clocks/div.})(8\text{div.})} = 75 \text{ KBytes/sec.}
\]

The data strobe and acknowledge timing are shown in Photo 2. The following values are determined from the trace.

- **Data Strobe pulse** = 500 nsec.
- **Acknowledge pulse** = 3.5 microsec.
- **Acknowledge delay** = 6.0 microsec.
- **Data Set up Time** = 5.0 microsec.
- **Data Hold Time** = 2.0 microsec.
Each of these values is within specification (13).

Timing for data transfer from 8051 internal RAM to the printer is shown in Photo 3 and Photo 4. When Busy goes inactive, the printer buffer (132 bytes) is filled. Then Busy goes active again during printing. The data transfer rate is calculated as follows:

\[
\text{Transfer Rate} = \frac{132 \text{ bytes}}{(20\mu\text{sec./clock})(20\text{clocks/div.})(5.8 \text{ div.})}
\]

\[
= 56.9 \text{ KBytes/sec.}
\]

Photo 5 shows the Input Prime pulse generated by the 8051 internal timer (T0). A 40 millisec pulse is desired and is verified as follows:

\[
\text{Input Prime} = (250\mu\text{sec./clock})(20\text{clocks/div.})(8\text{div.})
\]

\[
\text{Pulse Width} = 40 \text{ millisec.}
\]

The previously discussed test results are valid when In-Circuit Emulation (ICE51) is used with a 10 MHz clock. To further test the system an EPROM version (C8751-8) of the 8051 is used with the test circuitry (18, 19). Since this part is limited to 8 MHz operation, the 10 MHz oscillator is replaced
with a 7.3728 MHz oscillator. The previously used code for capturing the "acknowledge" pulse is modified to insure proper handshaking. This is accomplished by allowing the pulse to increment internal Timer0. Timer0 is used in mode 2 as an 8-bit counter with automatic reload. The counter is initialized to all ones (FFHex) so that the first occurrence of "acknowledge" will cause the counter to overflow, setting the Timer0 overflow flag (TO). When overflow occurs, the counter is automatically reloaded to the contents of TH0, (FFHex). Therefore, by clearing TO after an overflow detection (i.e. "acknowledge" received), the next occurrence of "acknowledge" will once again set TH0. The code for this algorithm can be found in Appendix B. With this version of code, the C8751-8 system has reliable operation when used with an oscillator that is greater than 5 MHz and less than 8 MHz. If less than 5 MHz is used, then "data strobe" (RD) is no longer within specification (i.e. "data strobe" > 1μsec.).
Photo 1. FIFO To Printer Transfer  
(1 MHz Clock, 20 Clocks/Div.)

Photo 2. FIFO To Printer Transfer  
(2 MHz Clock, 2 Clocks/Div.)
Photo 3. 8051 RAM To Printer Transfer  
(50 KHz Clock, 20 Clocks/Div.)

Photo 4. 8051 RAM To Printer Transfer  
(50 KHz Clock, 2 Clocks/Div.)
SELECT
DATA
STROBE
ACKNLG

BUSY
INPUT
PRIME
DATA1
DATA2

Photo 5. Input Prime Timing
(4 KHz Clock, 20 Clocks/Div.)
V. CONCLUSION

A demonstration of the 8051 microcontroller was accomplished. Its features and capabilities exceed those of the 8048. Additional memory, bit orientated instructions, faster execution, additional I/O ports, counters and interrupts are some more noteworthy features. The 8051 has a virtually limitless application spectrum. Its performance as a FIFO/Centronics controller was found to be quite acceptable. Even though some of its capabilities were not utilized in this application, fast execution time makes the 8051 more suitable than the 8048 if maximum data transfer rate is desired. This efficient code execution is achieved through use of the bit orientated instructions and the high speed oscillator. Since the tested application did not adequately evaluate the performance capabilities of the FIFO data transfer at the 16-bit data bus, this evaluation is open to future study. Also, performance of the 10 MHz EPROM version (C8751) of the 8051 may be studied.
APPENDIX A

TEST CIRCUIT SCHEMATIC
SW16

DATA IN (O-)

FIFO/CENTRONIC CONTROLLER SCHEMATIC (SHEET 1 OF 2)
APPENDIX B

TEST PROGRAM FLOWCHARTS AND PROGRAM LISTING
INITIALIZATION

PULSE "INPUT PRIME" TO RESET PRINTER

EXECUTE REQUEST?

CALL SELGEN ("SELECT" GENERATION)

GO TO COMMAND DECODE
COMMAND DECODE

FIFO OUTPUT READY?

NO

YES

SHIFT OUT LOWER BYTE AND SAVE IN R2

SHIFT OUT UPPER BYTE AND SAVE IN R3

BYTE TRANSFER COMMAND?

NO

YES

ZERO REPEITION?

YES

CALL BYTE TRANSFER

CALL BYTE SAVE

CALL TRANSFER FROM RAM

WAIT FOR NEXT EXECUTE REQUEST

WAIT FOR NEXT EXECUTE REQUEST
BYTE TRANSFER

FIFO EMPTY?

NO

SHIFT OUT LOWER BYTE

GENERATE "DATA STROBE" USING MOVX

SHIFT OUT UPPER BYTE

"ACKNOWLEDGE" RECEIVED?

YES

GENERATE "DATA STROBE" USING MOVX

NO

FIFO EMPTY?

YES

RETURN
BYTE SAVE

SET STARTING RAM ADDRESS TO 30HEX

FIFO EMPTY?

YES

SHIFT OUT LOWER BYTE AND SAVE IN RAM

INCREMENT ADDRESS

SHIFT OUT UPPER BYTE AND SAVE IN RAM

INCREMENT ADDRESS

RAM FULL?

NO

RETURN

YES

NO
TRANSFER FROM RAM

SET STARTING RAM ADDRESS TO 30HEX

END OF STORED BYTES?

YES

OUTPUT CONTENTS OF RAM TO PRINTER

GENERATE "DATA STROBE"

INCREMENT RAM ADDRESS

"ACKNOWLEDGE" RECEIVED?

YES

LAST RAM LOCATION?

YES

RETURN

NO

END REPETITION?

NO

DECREMENT REPETITION #

RETURN

YES
NIBBLE SAVE

SET STARTING RAM ADDRESS TO 30HEX

FIFO EMPTY ?

YES

SHIFT OUT LOWER BYTE

CONVERT LOWER NIBBLE AND SAVE IN RAM

INCREMENT RAM ADDRESS

RETURN

NO

NO

RAM FULL ?

YES

INCREMENT RAM ADDRESS

CONVERT UPPER NIBBLE AND SAVE IN RAM

INCREMENT RAM ADDRESS

INCREMENT RAM ADDRESS

CONVERT LOWER NIBBLE AND SAVE IN RAM

SHIFT OUT UPPER BYTE
TEST
GENERATION

SET STARTING
RAM ADDRESS
TO 30HEX

SET STARTING
CHARACTER
TO 20 HEX (¥)

SAVE CHARACTER
IN RAM

INCREMENT
RAM ADDRESS

CHARACTER = "Z"?

RETURN

INCREMENT TO NEXT
CHARACTER

NO

YES
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>TITLE: 8051 FIFO/CENTRONICS CONTROL (TEST)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>DATE: SPRING SEMESTER 1983</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>PROJECT: EEL 6918 (GRADUATE RESEARCH)</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>AUTHOR: G. C. SCHAFFNER</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>00A0</td>
<td>18</td>
<td>BUSY BIT P2.0  BUSY FROM PRINTER (LOW TRUE)</td>
</tr>
<tr>
<td>19</td>
<td>00A1</td>
<td>19</td>
<td>PE BIT P2.1  PAPER EMPTY (LOW TRUE)</td>
</tr>
<tr>
<td>20</td>
<td>00A2</td>
<td>20</td>
<td>ORDY BIT P2.2  FIFO OUTPUT READY (LOW TRUE)</td>
</tr>
<tr>
<td>21</td>
<td>00A3</td>
<td>21</td>
<td>UBDK BIT P2.3  FIFO UPPER BYTE OUTPUT READY (LOW TRUE)</td>
</tr>
<tr>
<td>22</td>
<td>00B1</td>
<td>22</td>
<td>SELECT BIT P3.1  SELECT FROM PRINTER (LOW TRUE)</td>
</tr>
<tr>
<td>23</td>
<td>00B2</td>
<td>23</td>
<td>EX BIT P3.2  EXECUTE REQUEST (LOW TRUE)</td>
</tr>
<tr>
<td>24</td>
<td>00B3</td>
<td>24</td>
<td>FAULT BIT P3.3  FAULT FROM PRINTER (LOW TRUE)</td>
</tr>
<tr>
<td>25</td>
<td>00B4</td>
<td>25</td>
<td>ACKNLG BIT P3.4  ACKNOWLEDGE FROM PRINTER (HIGH TRUE)</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>00A4</td>
<td>30</td>
<td>LBSD BIT P2.4  LOWER BYTE SHIFT OUT (LOW TRUE)</td>
</tr>
<tr>
<td>31</td>
<td>00A5</td>
<td>31</td>
<td>UBSD BIT P2.5  UPPER BYTE SHIFT OUT (LOW TRUE)</td>
</tr>
<tr>
<td>32</td>
<td>00B0</td>
<td>32</td>
<td>INPI BIT P3.0  INPUT PRIME TO PRINTER (LOW TRUE)</td>
</tr>
<tr>
<td>33</td>
<td>00B6</td>
<td>33</td>
<td>STRB BIT P3.6  DATA STROBE TO PRINTER (LOW TRUE)</td>
</tr>
<tr>
<td>34</td>
<td></td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>36+1</td>
<td></td>
<td>36</td>
<td>$EJECT</td>
</tr>
</tbody>
</table>
LDC OBJ LINE SOURCE

37 ;********************************************
38 ;********************************************
39 ;**********************************
40 ;**********************************
41 ;**********************************
42 ;**********************************
43 ;**********************************
44 ;
45 ;
46 ; ORG 00H
47 ; AJMP INIT ;GO TO INIT AT RESET
48 ;
49 ;
50 ;********************************** INTO INTERRUPT SERVICE ROUTINE ***************
51 ;
52 ; ORG 03H ;EXECUTE INTERRUPT ROUTINE
53 ; CLR EA ;DISABLE INTERRUPTS
54 ; CLR EX0 ;DISABLE EXTERNAL INTERRUPT 0
55 ; AJMP CONT ;CONTINUE
56 ;
57 ;
58 ;********************************** TIMER 0 INTERRUPT SERVICE ROUTINE ***************
59 ;
60 ; ORG 0BH ;40mSEC TIMER ROUTINE
61 ; CLR EA ;DISABLE INTERRUPTS
62 ; CLR ET0 ;DISABLE TIMER 0 INTERRUPT
63 ; CLR TR0 ;DISABLE TIMER 0
64 ; AJMP EXEC ;WAIT FOR EXECUTE REQUEST
65 ;
66 ;
67 ;********************************** INT1 INTERRUPT SERVICE ROUTINE ***************
68 ;
69 ; ORG 13H ;"FAULT" INTERRUPT ROUTINE
70 ; CLR EA ;DISABLE INTERRUPTS
71 ; AJMP INTRET ;GO TO INTRET
72 ;
73 ;
74 ;********************************** TIMER 1 INTERRUPT SERVICE ROUTINE ***************
75 ;
76 ; ORG 1BH ;75mSEC TIMER ROUTINE
77 ; CLR EA ;DISABLE INTERRUPTS
78 ; CLR ET1 ;DISABLE TIMER 1 INTERRUPT
79 ; CLR TR1 ;DISABLE TIMER 1
80 ; AJMP CONT ;CONTINUE
81 ;
82 ;
83 +1 $EJECT
LDC OBJ
LINE
SOURCE

84  ;INITIALIZATION: This routine sets up I/O Ports; awaits execution request, and resets the printer.

85  

86  

87  

88  

89  

90  

0043  ORG 43H
0044  INIT: MOV A, #0FFH
0045  MOV P1, A
0046  MOV P2, A
0047  MOV P3, A
0048  MOV TH0, #11H
0049  CLR INPRI
0050  MOV TH0, #07H
0051  MOV TL0, #0CH
0052  SETB ET0
0053  SETB EA
0054  SETB TR0
0055  SJMP $; WAIT FOR TIMEOUT
0056  D2A9
0057  D2AF
0058  D28C
0059  80FE
005A  D280
005B  758107
005C  9006B
005D  C082
005E  C083
005F  32
0060  32D
0061  D2AB
0062  D2AF
0063  80FE
0064  758107
0065  9007C
0066  C082
0067  C083
0068  32
0069  3120
006A  D2AF
006B  0182
006C  +1; $EJECT

0057  SETB EA
0058  JENABLE INTERRUPTS
0059  SETB TR0
0060  JSTART TIMER 0
0061  MOV SP, #07H
0062  JRESET STACK POINTER
0063  MOV DPTR, #RET1
0064  JSET UP NEW INTERRUPT RETURN
0065  PUSH DPL
0066  PUSH DPH
0067  RETI
0068  JRETURN FROM INTERRUPT
0069  SETB EX0
006A  JENABLE EXECUTE INTERRUPT
006B  SETB EA
006C  JENABLE INTERRUPTS
006D  SJMP $
006E  JWAIT FOR EXECUTE REQUEST
006F  CON: MOV SP, #07H
0070  JRESET STACK POINTER
0071  MOV DPTR, #RET2
0072  JSET UP NEW INTERRUPT RETURN
0073  PUSH DPL
0074  PUSH DPH
0075  RETI
0076  ACALL SELGEN
0077  JGENERATE "SELECT"
0078  SETB EA
0079  JENABLE INTERRUPTS
007A  AJMP CMD
007B  JGO TO COMMAND DECODE
007C  +1; $EJECT
**SOURCE**

```assembly
0082 20A2FD
0085 C2A4
0087 AA90
0089 D2A4
008B AB90
008D 5100
0091 BA0206
0094 BB0004
0097 3166
0099 01B5
009B 314C
009D 5100
009F 01B5
00A1 BA0206
00A4 318C
00A6 5100
00A8 01B5
00AA BA0306
00AD 5128
00AF 5100
00B1 01B5
00B3 3166
00B5 016B
00B7 016B
00B9 016B
00BB 016B
00BD 016B
00BE 016B
00BF 016B
126 ;*******************************************************************************
127 ;**
128 ;** COMMAND DECODE: This routine removes one
129 ;** 16-bit word from the FIFO, decodes the lower
130 ;** byte for desired command (01Hex- Byte Trans-
131 ;** fer; 02Hex-Nibble Transfer; or 03Hex- Test
132 ;** Generation); then the upper byte contains
133 ;** the number of repetitions for transfers.
134 ;**
135 ;*******************************************************************************
136 ;
137 ;
138 CMD: JB ORUT, $; WAIT FOR FIFO READY
139 CLR LBSO; SHIFT OUT LOWER BYTE
140 MOV R2, P1; SAVE LOWER BYTE IN R2
141 SETB LBSO
142 CLR UBSO; SHIFT OUT UPPER BYTE
143 MOV R3, P1; SAVE UPPER BYTE IN R3
144 SETB UBSO
145 ;
146 CJNE R2, #01H, NIBCK; JMP IF NOT BYTE TRANSFER
147 CJNE R3, #00H, BREP; JMP IF REPEATED TRANSFER
148 ACALL BYTE; CALL BYTE TRANSFER SUBROUTINE
149 AJMP STOP
150 BREP: ACALL BSAVE; CALL BYTE SAVE SUBROUTINE
151 ACALL TRAM; CALL TRANSFER RAM SUBROUTINE
152 AJMP STOP
153 ;
154 NIBCK: CJNE R2, #02H, TESTCK; JMP IF NOT NIBBLE TRANSFER
155 ACALL NIB; CALL NIBBLE TRANSFER SUBROUTINE
156 ACALL TRAM; CALL TRANSFER RAM SUBROUTINE
157 AJMP STOP
158 ;
159 TESTCK: CJNE R2, #03H, NOCMD; JMP IN NOT TEST REQUEST
160 ACALL TEST; CALL TEST GENERATION SUBROUTINE
161 ACALL TRAM; CALL TRANSFER RAM SUBROUTINE
162 AJMP STOP
163 ;
164 NOCMD: ACALL BYTE; IF INVALID COMMAND; TRANSFER BYTES
165 STOP: AJMP RET1; GO WAIT FOR NEXT COMMAND
166 ;
167 ;
168 +1 $EJECT
```
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>169</td>
<td></td>
<td></td>
<td>ORG   100H</td>
</tr>
<tr>
<td>170</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>171</td>
<td></td>
<td></td>
<td>###### CHECK FIFO STATUS FOR &quot;FAULT&quot; INTERRUPT ######</td>
</tr>
<tr>
<td>172</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>0100 D083</td>
<td>173</td>
<td></td>
<td>INTRRET: POP DPH</td>
</tr>
<tr>
<td>0102 D082</td>
<td>174</td>
<td></td>
<td>POP DPL</td>
</tr>
<tr>
<td>0104 A8B2</td>
<td>175</td>
<td></td>
<td>MOV R4,DPL</td>
</tr>
<tr>
<td>0106 ADB3</td>
<td>176</td>
<td></td>
<td>MOV R5,DPL</td>
</tr>
<tr>
<td>010B 900110</td>
<td>177</td>
<td></td>
<td>MOV DPTR,RET3</td>
</tr>
<tr>
<td>010B CO82</td>
<td>178</td>
<td></td>
<td>PUSH DPL</td>
</tr>
<tr>
<td>010D CO83</td>
<td>179</td>
<td></td>
<td>PUSH DPH</td>
</tr>
<tr>
<td>010F 32</td>
<td>180</td>
<td></td>
<td>RET</td>
</tr>
<tr>
<td>0110 BD0202</td>
<td>181</td>
<td></td>
<td>RET3: CJNE R5,$02H,DI</td>
</tr>
<tr>
<td>0113 4123</td>
<td>182</td>
<td></td>
<td>AJMP IROM</td>
</tr>
<tr>
<td>0115 90016E</td>
<td>183</td>
<td></td>
<td>BI: MOV DPTR,$NEXT</td>
</tr>
<tr>
<td>011B E582</td>
<td>184</td>
<td></td>
<td>MOV A,DPL</td>
</tr>
<tr>
<td>011A C3</td>
<td>185</td>
<td></td>
<td>CLR C</td>
</tr>
<tr>
<td>011B 9C</td>
<td>186</td>
<td></td>
<td>SUBB A,R4</td>
</tr>
<tr>
<td>011C 5048</td>
<td>187</td>
<td></td>
<td>JNC BYTE</td>
</tr>
<tr>
<td>011E 900179</td>
<td>188</td>
<td></td>
<td>MOV DPTR,$CHECK</td>
</tr>
<tr>
<td>0121 E582</td>
<td>189</td>
<td></td>
<td>MOV A,DPL</td>
</tr>
<tr>
<td>0123 C3</td>
<td>190</td>
<td></td>
<td>CLR C</td>
</tr>
<tr>
<td>0124 9C</td>
<td>191</td>
<td></td>
<td>SUBB A,R4</td>
</tr>
<tr>
<td>0125 403F</td>
<td>192</td>
<td></td>
<td>JC BYTE</td>
</tr>
<tr>
<td>0127 C2A5</td>
<td>193</td>
<td></td>
<td>CLR UBSO</td>
</tr>
<tr>
<td>0129 D2AF</td>
<td>194</td>
<td></td>
<td>SETB EA</td>
</tr>
<tr>
<td>012B 217B</td>
<td>195</td>
<td></td>
<td>AJMP UPPER</td>
</tr>
<tr>
<td>196</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>197</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>198</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>199</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>200</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>201</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>202</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>203</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>012D 759011</td>
<td>204</td>
<td></td>
<td>Selenen: MOV P1,$11H</td>
</tr>
<tr>
<td>0130 F2</td>
<td>205</td>
<td></td>
<td>MOVX @RO,A</td>
</tr>
<tr>
<td>0131 85FF8B</td>
<td>206</td>
<td></td>
<td>MOV TL1,OFFH</td>
</tr>
<tr>
<td>0134 8500BD</td>
<td>207</td>
<td></td>
<td>MOV TH1,OBH</td>
</tr>
<tr>
<td>0137 D2AB</td>
<td>208</td>
<td></td>
<td>SETB ET1</td>
</tr>
<tr>
<td>0139 D2AF</td>
<td>209</td>
<td></td>
<td>SETB EA</td>
</tr>
<tr>
<td>013B D2BE</td>
<td>210</td>
<td></td>
<td>SETB TR1</td>
</tr>
<tr>
<td>013D 2081FD</td>
<td>211</td>
<td></td>
<td>JB SELECT,$</td>
</tr>
<tr>
<td>0140 C2AF</td>
<td>212</td>
<td></td>
<td>CLR EA</td>
</tr>
<tr>
<td>0142 C2AB</td>
<td>213</td>
<td></td>
<td>CLR ET1</td>
</tr>
<tr>
<td>0144 C2BE</td>
<td>214</td>
<td></td>
<td>CLR TR1</td>
</tr>
<tr>
<td>0146 7590FF</td>
<td>215</td>
<td></td>
<td>MOV P1,$OFFH</td>
</tr>
<tr>
<td>0149 D2AA</td>
<td>216</td>
<td></td>
<td>SETB EX1</td>
</tr>
<tr>
<td>014B 22</td>
<td>217</td>
<td></td>
<td>RET</td>
</tr>
<tr>
<td>218 +1 $EJECT</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BYTE SAVE: This subroutine transfers 80 bytes from the FIFO to internal RAM locations 30Hex thru 79Hex.

EXIT VALUES: R1 = Final address of stored data in internal RAM.

BYTE SAVE: MOV R1,#30H ;Set starting RAM address to 30Hex
   JB ORDY/EMPTY ;Jump if FIFO empty

BYTE: SETB EA ;Enable interrupts
   JB ORDY,BEND ;Wait for FIFO ready
   CLK LSB0 ;Shift-out lower byte
   MOVX @RO,A ;Generate "DATA STROBE" (WR pulse)

NEXT: SETB LSB0 ;Disable lower byte
   CLR USB0 ;Shift-out upper byte
   JNB ACKNL6$ ;Wait for "ACKNOWLEDGE"
   JB ACKNL6$ ;

UPPER: MOVX @RO,A ;"DATA STROBE" for upper byte

CHECK: SETB USB0 ;Disable upper byte
   JB ORDY,BYTE ;If FIFO not ready go to CKFIFO
   CLR LSB0 ;Shift-out lower byte
   JNB ACKNL6$ ;Wait for "ACKNOWLEDGE"
   JB ACKNL6$ ;
   MOVX @RO,A ;"DATA STROBE" for lower byte

BEND: AJMP NEXT ;Send next byte

EJECT: CLR EA ;Disable interrupts
   RET

BYTE TRANSFER: This subroutine transfers bytes directly from the FIFO to the printer until the FIFO is empty.

BYTE: SETB EA ;Enable interrupts
   JB ORDY,BEND ;Wait for FIFO ready
   CLK LSB0 ;Shift-out lower byte
   MOVX @RO,A ;Generate "DATA STROBE" (WR pulse)

NEXT: SETB LSB0 ;Disable lower byte
   CLR USB0 ;Shift-out upper byte
   JNB ACKNL6$ ;Wait for "ACKNOWLEDGE"
   JB ACKNL6$ ;

UPPER: MOVX @RO,A ;"DATA STROBE" for upper byte

CHECK: SETB USB0 ;Disable upper byte
   JB ORDY,BYTE ;If FIFO not ready go to CKFIFO
   CLR LSB0 ;Shift-out lower byte
   JNB ACKNL6$ ;Wait for "ACKNOWLEDGE"
   JB ACKNL6$ ;
   MOVX @RO,A ;"DATA STROBE" for lower byte

BEND: AJMP NEXT ;Send next byte

EJECT: CLR EA ;Disable interrupts
   RET
NIBBLE SAVE: This subroutine receives bytes from the FIFO, converts the nibbles to Hex/ASCII equivalent, and saves the corresponding byte in internal RAM.

Exit values: $R1 = \text{Address of last data in internal RAM to be transferred.}$

;*******************************************************************************
271 ;
272 ;
273 ; Nibble save: This subroutine receives bytes from the FIFO, converts the
274 ; nibbles to Hex/ASCII equivalent, and saves the corresponding byte in
275 ; internal RAM (locations 30Hex to 79Hex).
276 ;
277 ; Exit values: $R1 = \text{Address of last data in internal RAM to be transferred.}$
278 ;
279 ;*******************************************************************************

01BC 7F01
01BE 7930
0190 20422D
0193 1F
0194 C2A4
0196 AC90
019B D2A4
019A 0F
019B EC
019C 540F
019E 2490
01A0 D4
01A1 3440
01A3 D4
01A4 F7
01A5 09
01A6 EC
01A7 C4
01AB S40F
01AA 2490
01AC D4
01AD 3440
01AF D4
01B0 F7
01B1 09
01B2 DF09
01B4 C2A5
01B6 AC90
01BB D2A5
01BA 0F
01BB 219A
01BD B98080
01C0 22

;*******************************************************************************
284 ;
285 NIB: MOV R7,#01H ;Upper or lower byte status flag
286 MOV R1,#30H ;Initialize address location
287 ;
288 NXTNIB: JB ORDY,DONE ;Jump if FIFO empty
289 ;
290 CONV: INC R7 ;Status incremented
291 MOV A,R4 ;Save lower byte in R4
292 ;
293 ;
294 ;
295 ;
296 ;
297 ;
298 ;
299 ;
300 ;
301 ;
302 ;
303 ;
304 ;
305 ;
306 ;
307 ;
308 ;
309 ;
310 ;
311 ;
312 ;
313 ;
314 ;
315 ;
316 ;
317 ;
318 ;
319 +1 $EJECT
TRANSFER RAM: This subroutine transfers data from internal RAM (locations 30Hex to [R1]) to the printer.

ENTRY VALUES: R1 = Address of terminating location.
R3 = Number of repetitions RAM data is to be transferred.

DATA IS TO BE TRANSFERRED.

TRAM: MOV A,R1 ;SAVE TERMINATING ADDRESS
       MOV R7,A ;IN R7
       INC R3 ;ENABLE INTERRUPTS
       SETB EA ;SET UP STARTING ADDRESS
       MOV R1,#30H ;
       MOV A,R1 ;
       XRL A,R7 ;COMPARE TERMINATING ADD. TO START ADD.
       JZ ENDBS ;IF SAME, JUMP TO ENDBS
       MOVX @R0,A ;OUTPUT RAM CONTENTS TO PRINTER
       MOVX @T<01A,R1 ;
       MOVX @R1,A ;
       XRL A,R7 ;COMPARE CURRENT ADD. TO TERMINATING ADD.
       JZ ENDBS ;IF SAME, JUMP TO ENDBS
       JB ACKNLG$ ;WAIT FOR "ACKNOWLEDGE" FROM PRINTER
       JB ACKNLG$ ;
       INTR: XRL A,R7 ;COMPARE CURRENT ADD. TO TERMINATING ADD.
       JNZ NEXTB ;JUMP IF NOT EQUAL
       INC R1 ;INCREMENT ADDRESS
       MOV P1,#R1 ;OUTPUT NEXT RAM CONTENTS TO PRINTER
       MOV A,R1 ;
       MOVX @R1,A ;
       JNB ACKNL61S ;COMPARISON CURRENT ADD. TO TERMINATING ADD.
       JB ACKNLG$ ;IF NOT EQUAL 0 REPEAT TRANSFER
       MDV R1,#30H ;RESET STARTING ADDRESS
       INC R3 ;DECREMENT REPETITION
       DEC R3 ;
       CJNE R3,#00H,REPT ;IF 0 NOT EQUAL 0 REPEAT TRANSFER
       AJMP ENDBS ;IF = 0 GO TO ENDBS
       AJMP ENDBS ;LOAD ADDRESS INTO ACC.
       SETB EA ;ENABLE INTERRUPTS
       AJMP INTR ;DISABLE INTERRUPTS
       CLR EA ;
       RET ;
       ;
       ;
       RET
```
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>365</td>
<td></td>
<td>62</td>
<td>62</td>
</tr>
<tr>
<td>366</td>
<td></td>
<td></td>
<td>;******</td>
</tr>
<tr>
<td>367</td>
<td></td>
<td></td>
<td>;******</td>
</tr>
<tr>
<td>368</td>
<td></td>
<td></td>
<td>;****** TEST GENERATION: This subroutine generates ******</td>
</tr>
<tr>
<td>369</td>
<td></td>
<td></td>
<td>;****** 59 characters to be printed (20Hex to 5AHex) ******</td>
</tr>
<tr>
<td>370</td>
<td></td>
<td></td>
<td>;****** and saves them in internal RAM (locations 30 ******</td>
</tr>
<tr>
<td>371</td>
<td></td>
<td></td>
<td>;****** Hex to 6AHex). ******</td>
</tr>
<tr>
<td>372</td>
<td></td>
<td></td>
<td>;******</td>
</tr>
<tr>
<td>373</td>
<td></td>
<td></td>
<td>;******</td>
</tr>
<tr>
<td>374</td>
<td></td>
<td></td>
<td>;******</td>
</tr>
<tr>
<td>375</td>
<td></td>
<td></td>
<td>;******</td>
</tr>
<tr>
<td>376</td>
<td>022B</td>
<td>7930</td>
<td>TEST: MOV R1,#30H ;SET STARTING ADDRESS</td>
</tr>
<tr>
<td>377</td>
<td>022D</td>
<td>7D20</td>
<td>MOV R5,#20H ;STARTING CHARACTER=20Hex( )</td>
</tr>
<tr>
<td>378</td>
<td>022F</td>
<td>7E5A</td>
<td>MOV R6,#5AH ;ENDING CHARACTER=5AHex(Z)</td>
</tr>
<tr>
<td>379</td>
<td>0231</td>
<td>ED</td>
<td>NXTST: MOV A,$R5 ;SAVE CHARACTER AT [R1]</td>
</tr>
<tr>
<td>380</td>
<td>0232</td>
<td>F7</td>
<td>MOV @R1,A ;COMPARE LAST CHARACTER TO &quot;Z&quot;</td>
</tr>
<tr>
<td>381</td>
<td>0233</td>
<td>6E</td>
<td>XRL A,$R6 ;JUMP IF &quot;Z&quot;</td>
</tr>
<tr>
<td>382</td>
<td>0234</td>
<td>6004</td>
<td>INC R1 ;INCREMENT CHARACTER</td>
</tr>
<tr>
<td>383</td>
<td>0236</td>
<td>09</td>
<td>INC R5 ;GO TO NEXT CHARACTER</td>
</tr>
<tr>
<td>384</td>
<td>0237</td>
<td>0D</td>
<td>AHMP NXTST ;SAVE NEXT CHARACTER</td>
</tr>
<tr>
<td>385</td>
<td>0238</td>
<td>4131</td>
<td>TEND: MOV R1,#6BH ;PLACE TERMINATING ADDRESS INTO R1</td>
</tr>
<tr>
<td>386</td>
<td>0239</td>
<td>7961</td>
<td></td>
</tr>
<tr>
<td>387</td>
<td>023A</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>388</td>
<td>023C</td>
<td>22</td>
<td></td>
</tr>
<tr>
<td>389</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>390</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ASSEMBLY COMPLETE, NO ERRORS FOUND
```
BYTE TRANSFER: This subroutine transfers bytes directly from the FIFO to the printer until the FIFO is empty.

BYTE TRANSFER ROUTINE FOR C8751-8 (5-8 MHz)
TRANSFER RAM: This subroutine transfers data from internal RAM (locations 30Hex to CR1) to the printer.

ENTRY VALUES: R1 = Address of terminating location.

R3 = Number of repetitions

RAM data is to be transferred.

; TRANSFER RAM:
    MOV A,R1            ;Save terminating address
    MOV R7,A            ;IN R7
    INC R3              ;Set timer as 8-bit counter
    MOV TMOD,#16H       ;Set low byte of counter to FF
    MOV TL0,#0FFH       ;Set reload value to FF
    MOV TH0,#0FFH       ;Clear overflow bit
    SETB TR0            ;Enable counter
    SETB EA             ;Enable interrupts
    MOV R1,#30H         ;Set up starting address
    MOV A,R1
    XRL A,R7            ;Compare terminating add. to start add.
    JZ ENDBS            ;If same, jump to ENDBS
    INC R9              ;Subroutine transfer
    MOV P1<,R1          ;Output RAM contents to printer
    MOV X,#00A          ;Generate "data strobe" (NR)
    INC R1              ;Increment address
    MOV P1<,R1          ;Output next RAM contents to printer
    MOV A,R1
    JNB TF0<,#          ;Wait for "acknowledge" from printer
    CLR TF0             ;Clear overflow flag
    XRL A,R7            ;Compare current add. to terminate add.
    JNZ NEXTB          ;Jump if not equal
    MOV R1,#30H         ;Reset starting address
    DEC R3              ;Decrement repetition
    JNC NEXTB         ;If not equal, 0 repeat transfer
    CJE R3,#300H,REPT   ;If # not equal 0 repeat transfer
    AJMP ENDBS          ;If = 0 go to ENDBS
    IRAM: SETB EA       ;Enable interrupts
    MOV A,R1            ;Load address into ACC.
    AJMP INTR
    ENDBS: CLR EA       ;Disable interrupts
    RET

RAM TRANSFER ROUTINE FOR C8751-8 (5-8 MHz)
APPENDIX C

FIFO SPECIFICATION
The 67401 is an expandable high speed First-In First-Out (FIFO) memory organized 64 words by 4-bits. A 10 MHz data rate allows usage in high speed tape or disc controllers and communication buffer applications.

**67401 BLOCK DIAGRAM**
67401 PIN CONFIGURATION

Functional Description

Data Input

Data is entered into the FIFO on DO-D3 inputs. To enter data the Input Ready (IR) should be HIGH, indicating that the first location is ready to accept data. Data then present at the four inputs is entered into the first location when the Shift In (SI) is brought HIGH. A SI HIGH signal causes the IR to go low. Data remains at the first location until SI is brought LOW. When SI is brought LOW and the FIFO is not full, IR will go HIGH, indicating that more room is available. Simultaneously, data will propagate to the second location and continue shifting until it reaches the output stage or a full location. If the memory is full, IR will remain LOW.
Data Transfer

Once data is entered into the second cell, the transfer of any full cell to the adjacent empty cell is automatic, activated by an on-chip control. Thus data will stack up at the end of the device while empty locations will "bubble" to the front. The maximum time required for the first data to travel from input to the output of a previously empty device is 3 microseconds.

Data Output

Data is read from the 00-03 outputs. When data is shifted to the output stage, Output Ready (OR) goes HIGH, indicating the presence of valid data. When the OR is HIGH, data may be shifted out by bringing the Shift Out (SO) HIGH. A HIGH signal at SO causes the OR to go LOW. Valid data is maintained while the SO is HIGH. When SO is brought LOW the upstream data, provided that stage has valid data, is shifted to the output stage. When new valid data is shifted to the output stage, OR goes HIGH. If the FIFO is emptied, OR stays LOW, and 00-03 remains as before (i. e. data does not change if FIFO is empty).
Input Ready and Output Ready may also be used as status signals indicating that the FIFO is completely full (IR stays LOW for at least 3 microsec.) or completely empty (OR stays LOW for at least 3 microsec.).
### Switching Characteristics

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>PARAMETER</th>
<th>MIN</th>
<th>MAX</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>fi</td>
<td>Shift in rate</td>
<td>10</td>
<td></td>
<td>MHz</td>
</tr>
<tr>
<td>T1</td>
<td>Shift in HIGH time</td>
<td>35</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T2</td>
<td>Shift in LOW time</td>
<td>35</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T3</td>
<td>Shift in to input ready LOW</td>
<td>45</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T4</td>
<td>Shift in to input ready HIGH</td>
<td>45</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T5</td>
<td>Input data set up</td>
<td>5</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T6</td>
<td>Input data hold time</td>
<td>45</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>fo</td>
<td>Shift Out rate</td>
<td>10</td>
<td></td>
<td>MHz</td>
</tr>
<tr>
<td>T7</td>
<td>Shift Out HIGH time</td>
<td>35</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T8</td>
<td>Shift Out LOW time</td>
<td>35</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T9</td>
<td>Shift Out to Output Ready LOW</td>
<td>55</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T10</td>
<td>Shift Out to Output Ready HIGH</td>
<td>55</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T11</td>
<td>Output data delay</td>
<td>10</td>
<td>55</td>
<td>ns</td>
</tr>
<tr>
<td>T12</td>
<td>Data throughput time</td>
<td>3</td>
<td></td>
<td>µs</td>
</tr>
<tr>
<td>T13</td>
<td>Master Reset pulse</td>
<td>35</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T14</td>
<td>Master Reset to OR LOW</td>
<td>60</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T15</td>
<td>Master Reset to IR HIGH</td>
<td>60</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T16</td>
<td>Master Reset to SI</td>
<td>35</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T17</td>
<td>Input Ready pulse HIGH</td>
<td>20</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>T18</td>
<td>Output Ready pulse HIGH</td>
<td>20</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>
APPENDIX D

CENTRONICS - COMPATIBLE INTERFACE SPECIFICATION
A Centronics - Compatible Interface option is available on the Data products M - 200 matrix printer. The option is configured to be electrically and mechanically compatible with printer controllers designed for the Centronics 101A printer.

ASCII codes are received in bit parallel format and data transfer between the user and printer is on a strobe/acknowledge format. This allows the user to control the data transfer rate at a maximum of 75 kilobytes per second.

The Centronics interface is designed to accommodate data transmission over 49 feet maximum. Signals between the user and printer should be transmitted over twisted pair wires. Recommended receiver and transmitter circuits are shown below.
A logic "1" must be greater than +2.4VDC and less than +5.0VDC. A logic "0" must be greater than 0.0VDC and less than +0.4VDC.

An adaptor is used to provide mechanical compatibility with the Centronics-type 36pin connector. The connector is Amphenol part number 57-40360. The mating connector is Amphenol part number 57-30360.

The interface lines between the printer and user, along with the corresponding pin assignments, are listed below.

<table>
<thead>
<tr>
<th>SIGNAL</th>
<th>PIN</th>
<th>SIGNAL</th>
<th>PIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATASTROBE</td>
<td>1</td>
<td>ACKNLG</td>
<td>10</td>
</tr>
<tr>
<td>DATASTROBE RTN</td>
<td>19</td>
<td>ACKNLG RTN</td>
<td>28</td>
</tr>
<tr>
<td>DATA1</td>
<td>2</td>
<td>BUSY</td>
<td>11</td>
</tr>
<tr>
<td>DATA1 RTN</td>
<td>20</td>
<td>BUSY RTN</td>
<td>29</td>
</tr>
<tr>
<td>DATA2</td>
<td>3</td>
<td>PE</td>
<td>12</td>
</tr>
<tr>
<td>DATA2 RTN</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA3</td>
<td>4</td>
<td>SELECT</td>
<td>13</td>
</tr>
<tr>
<td>DATA3 RTN</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA4</td>
<td>5</td>
<td>OSCXT</td>
<td>15</td>
</tr>
<tr>
<td>DATA4 RTN</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATA5</td>
<td>6</td>
<td>INPUT PRIME</td>
<td>31</td>
</tr>
<tr>
<td>DATA5 RTN</td>
<td>24</td>
<td>INPUT PRIME RTN</td>
<td>30</td>
</tr>
<tr>
<td>DATA6</td>
<td>7</td>
<td>FAULT</td>
<td>32</td>
</tr>
<tr>
<td>DATA6 RTN</td>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Line</td>
<td>Value</td>
<td>Description</td>
<td>Pin Numbers</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td>---------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>DATA7</td>
<td>8</td>
<td>+5V</td>
<td>18</td>
</tr>
<tr>
<td>RTN</td>
<td>26</td>
<td>0V</td>
<td>14,16</td>
</tr>
<tr>
<td>DATA8</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RTN</td>
<td>27</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The interface signals are defined as follows:

**SELECT** - A printer generated signal which indicates that the printer has been selected. Before the printer can receive print or format data, it must be selected. The printer can be selected by depressing the ON-LINE switch, or by receiving a hex (11) via the data bus. When the printer has been selected an acknowledge pulse will be transmitted to the user after the Busy signal goes inactive. The printer can be deselected by again depressing the ON-LINE switch or by receiving a hex (13) via the data bus. Select is active in the logic "1" state.

**ACKNLG** - A printer generated signal which
acknowledges that the printer has received a data word. If the data word produces a busy condition the acknowledge signal will not be generated until the busy condition is reset. Acknowledge is active in the logic "0" state.

**DATA STROBE** - A user generated signal which defines when information on the data lines is stable and may be stored in the printer buffer. Data Strobe is active in the logic "0" state.

**BUSY** - A printer generated signal that indicates the printer is unable to receive print or format data. Busy is active in the logic "1" state.

**INPUT PRIME** - A user generated signal that will clear the printer buffer and initializes the interface logic. The Input Prime signal is asynchronous to the interface logic and is active
in the logic "0" state.

**FAULT** -
A printer generated signal indicating that one of the following faults has occurred:
(a) Printer is Out-Of-Paper.
(b) Shuttle is not moving.
(c) Printer is deselected.
Fault is active in the logic "0" state.

**OSCXT** -
(OSCILLATOR)
A printer generated signal that transmits a 100 KHz square wave to the user.

**PE** -
(PAPER EMPTY)
A printer generated signal that indicates the printer is Out-Of-Paper. Paper Empty is active in the logic "1" state.

When power is applied to the printer, all interface signals, except for "Busy", are set to their inactive state. The Busy signal is set active to indicate that the printer is unable to receive data.
When the auto line feed option is installed, the printer buffer is completely filled (132 standard character load or 219 condensed character load). Then the printer will automatically terminate the load, initiate the print cycle, and advance the form one line.

When the auto line feed option is not installed, the printer will not terminate automatically on either the 132nd or 220th character, but will continue to handshake until terminated by either a LF (0A Hex) or any vertical format command. Standard or condensed characters received after the 132nd or 219 characters respectively will be disregarded.

The interface handshaking signals operate in a pulse mode. Once the printer has been selected and there is no busy condition, the pulsed handshaking will operate as follows:

(a) The user will transmit a data strobe to the printer.

(b) The printer senses the active data strobe, and stores the data word into memory.

(c) For handshaking without a busy condition, the printer then senses the inactive data strobe, waits
during the acknowledge delay, and then issues an acknowledge pulse.

For handshaking with a busy condition, the busy signal will go active until the condition is reset, then the printer will issue an acknowledge pulse after the busy signal goes inactive.

(d) The user then senses the active acknowledge pulse, and can then transmit another data strobe to the printer.

Timing diagrams and restrictions are shown on the following pages.
A = 1.0 microsec. min.
B = 0.5 microsec. min.
C = 4 + 1.0 microsec.
D = Greater than zero

SELECT GENERATION VIA DATA BUS
SELECT

DATA STROBE

ACKNLG

DATA

BUSY

A = 1.0 microsec.min.  D = 4+ 1.0 microsec.
B = 0.5 microsec.min.  E = Greater than zero
C = 7+ 1.2 microsec.  F = 0.0 microsec.min.

= 1.0 microsec.max.
= 0.075 microsec.max.

DATA TRANSFER TIMING WITHOUT BUSY
SELECT  +I A DATA STROBE

DATA BUSY ACKNLG

DATA TRANSFER TIMING WITH BUSY

<table>
<thead>
<tr>
<th>Busy Condition</th>
<th>Busy Duration</th>
<th>Print + paper motion cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Deselect code</td>
<td>Until printer is selected</td>
<td>7.0±1.0 microsec.</td>
</tr>
<tr>
<td>2. CR W/O Auto Line</td>
<td>Print + paper motion cycle</td>
<td></td>
</tr>
<tr>
<td>3. CR W/ Auto Line</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. All Other Terminations</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LIST OF REFERENCES


15. ICE-51 In-Circuit Emulator Operation Instructions For ISIS-II Users, Santa Clara: Intel Corporation, 1981.


19. iPPS Intel PROM Programming Software Version 1.0 (Computer program), Santa Clara: Intel Corporation.