An Evaluation of the 8051 Microcontroller

George C. Schafner
University of Central Florida
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 INTR0 and INTR1 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).
<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:

```
CRL  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.

\[
\begin{array}{l}
\text{8048} \\
\text{MOV } A, \#0 \text{ACH} \\
\text{OUTL } P1, A \\
4 \text{ cycles } = 10.0 \text{ microsec.}
\end{array}
\quad
\begin{array}{l}
\text{8051} \\
\text{MOV } P1, \#0 \text{ACH} \\
\end{array}
\quad
\begin{array}{l}
2 \text{ cycles } = 2.0 \text{ microsec.}
\end{array}
\]

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>Last Byte</td>
<td>127th</td>
</tr>
<tr>
<td>To Be</td>
<td></td>
</tr>
<tr>
<td>Printed</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>64th Word</th>
<th>5th</th>
</tr>
</thead>
<tbody>
<tr>
<td>4th</td>
<td>3rd</td>
</tr>
<tr>
<td>2nd</td>
<td>1st Byte</td>
</tr>
<tr>
<td></td>
<td>To Be</td>
</tr>
<tr>
<td></td>
<td>Printed</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 \( \overline{WR} \) pulse width = (6) (P) - 100 nanoseconds, where "P" is the oscillator period. A 10 MHz oscillator is therefore needed for a 0.5 microsecond \( \overline{WR} \) 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)
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 is 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)
INITIALIZATION

PULSE "INPUT PRIME" TO RESET PRINTER

CALL SELGEN ("SELECT" GENERATION)

FIFO OUTPUT READY?

YES

"ACKNOWLEDGE" RECEIVED?

YES

GENERATE "DATA STROBE" PULSE USING MOVX INSTRUCTION

SHIFT OUT UPPER BYTE

NO

NO

SHIFT OUT LOWER BYTE

GENERATE "DATA STROBE" PULSE USING MOVX INSTRUCTION

"ACKNOWLEDGE" RECEIVED?

YES

NO

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.
Figure 9. Flowchart of reduced delay sequence
Figure 10. Flowchart for SELGEN ("Select" generation) subroutine
**INPUT PIN DECLARATIONS**

18  BUSY  BIT P2.0  ;BUSY FROM PRINTER (LOW TRUE)
19   PE   BIT P2.1  ;PAPER EMPTY (LOW TRUE)
20  ORDY  BIT P2.2  ;FIFO OUTPUT READY (LOW TRUE)
21  UBOK  BIT P2.3  ;FIFO UPPER BYTE OUTPUT READY (LOW TRUE)
22  SELECT BIT P3.1  ;SELECT FROM PRINTER (LOW TRUE)
23    EX   BIT P3.2  ;EXECUTE REQUEST (LOW TRUE)
24  FAULT BIT P3.3  ;FAULT FROM PRINTER (LOW TRUE)
25  ACKNLG BIT P3.4  ;ACKNOWLEDGE FROM PRINTER (HIGH TRUE)

**OUTPUT PIN DECLARATIONS**

28  ;
29  ;
30  LBSO  BIT P2.4  ;LOWER BYTE SHIFT OUT (LOW TRUE)
31  UBSO  BIT P2.5  ;UPPER BYTE SHIFT OUT (LOW TRUE)
32  IMPRI BIT P3.0  ;INPUT PRIME TO PRINTER (LOW TRUE)
33  STKB  BIT P3.6  ;DATA STROBE TO PRINTER (LOW TRUE)
34  ;
35  ;
36  +1  $EJECT
;**********************************************************************
;**                        INTERRUPT SERVICE ROUTINES                    **
;**********************************************************************

0000  ORG OOH
0000  AJMP INIT ;GO TO INIT AT RESET

0003  ORG 03H ;EXECUTE INTERRUPT ROUTINE
0003  CLR EA ;DISABLE INTERRUPTS
0005  CLR EX0 ;DISABLE EXTERNAL INTERRUPT 0
0007  AJMP CONT ;CONTINUE

000B  ORG 0BH ;40mSEC TIMER ROUTINE
000B  CLR EA ;DISABLE INTERRUPTS
000D  CLR ET0 ;DISABLE TIMER 0 INTERRUPT
000F  CLR TR0 ;DISABLE TIMER 0
0011  AJMP EXEC ;WAIT FOR EXECUTE REQUEST

0013  ORG 13H ;"FAULT" INTERRUPT ROUTINE
0013  CLR EA ;DISABLE INTERRUPTS
0015  AJMP INTRET ;GO TO INTRET

001B  ORG 1BH ;75mSEC TIMER ROUTINE
001B  CLR EA ;DISABLE INTERRUPTS
001D  CLR ET1 ;DISABLE TIMER 1 INTERRUPT
001F  CLR TR1 ;DISABLE TIMER 1
0021  AJMP CONT ;CONTINUE

83 +1 $EJECT
INITIALIZATION: This routine sets up I/O
Ports, awaits execution request, and resets
the printer.

ORG 43H
SET UP I/O PORTS

MOV A,$00FH
MOV P1A
MOV P2A
MOV P3A
MOV TMOD,#11H
CLR INPRI
MOV TLO,#0C9H
MOV THO,#07DH
SETB ETO
SETB EA
SETB TR0
SJMP $
MOV SP,#07H
PUSH DPL
PUSH DPH
RETI

SETB EX0
SETB EA
SJMP$
MOV SP,#07H
PUSH DPL
PUSH DPH
RET1:

CONT:
MOV SP,#07H
PUSH DPL
PUSH DPH
RET2:
ACALL SELGEN
AJMP CKFIFO

+1 $EJECT
<table>
<thead>
<tr>
<th>LOC OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>124</td>
<td>;**** CHECK FIFO STATUS FOR &quot;FAULT&quot; INTERRUPT *******</td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>0082 D083</td>
<td>126</td>
<td>RETI: POP DPH  ; REMOVE ADDRESS WHERE INTERRUPT OCCURED</td>
</tr>
<tr>
<td>0084 D082</td>
<td>127</td>
<td>POP DPL  ; LOWER BYTE OF ADDRESS IN DPL</td>
</tr>
<tr>
<td>0086 ACB2</td>
<td>128</td>
<td>MOV R4,DPL  ; SAVE LOWEK BYTE IN R4</td>
</tr>
<tr>
<td>0088 900090</td>
<td>129</td>
<td>MOV DPTR,#RET3  ; SET UP NEW INTERRUPT RETURN</td>
</tr>
<tr>
<td>008B C082</td>
<td>130</td>
<td>PUSH DPL</td>
</tr>
<tr>
<td>008D CO83</td>
<td>131</td>
<td>PUSH DPH</td>
</tr>
<tr>
<td>008F 32</td>
<td>132</td>
<td>RETI</td>
</tr>
<tr>
<td>0090 9000AE</td>
<td>133</td>
<td>RET3: MOV DPTR,#NEXT  ; FIND WHERE INTERRUPT OCCURED</td>
</tr>
<tr>
<td>0093 E5B2</td>
<td>134</td>
<td>MOV A,DPL</td>
</tr>
<tr>
<td>0095 C3</td>
<td>135</td>
<td>CLR C</td>
</tr>
<tr>
<td>0096 9C</td>
<td>136</td>
<td>SUBB A,R4  ; COMPARE &quot;NEXT&quot; ADD. TO INTERRUPT ADD.</td>
</tr>
<tr>
<td>0097 500F</td>
<td>137</td>
<td>JNC CKFIFO  ; JUMP IF INTERRUPT ADD. (= &quot;NEXT&quot;</td>
</tr>
<tr>
<td>0099 9000B9</td>
<td>138</td>
<td>MOV DPTR,#CHECK</td>
</tr>
<tr>
<td>009C E5B2</td>
<td>139</td>
<td>MOV A,DPL</td>
</tr>
<tr>
<td>009E C3</td>
<td>140</td>
<td>CLR C</td>
</tr>
<tr>
<td>009F 9C</td>
<td>141</td>
<td>SUBB A,R4  ; COMPARE &quot;CHECK&quot; ADD. TO INTERRUPT ADD.</td>
</tr>
<tr>
<td>00A0 4006</td>
<td>142</td>
<td>JC CKFIFO  ; JUMP IF INTERRUPT ADD. ) &quot;CHECK&quot;</td>
</tr>
<tr>
<td>00A2 C2A5</td>
<td>143</td>
<td>CLR UB50  ; SHIft OUT UPPER BYTE</td>
</tr>
<tr>
<td>00A4 D2AF</td>
<td>144</td>
<td>SETB EA  ; ENABLE INTERRUPTS</td>
</tr>
<tr>
<td>00A6 01B8</td>
<td>145</td>
<td>AJMP UPPER  ; SEND UPPER BYTE</td>
</tr>
<tr>
<td>146</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>147</td>
<td>;</td>
<td></td>
</tr>
<tr>
<td>148 +1</td>
<td>$EJECT</td>
<td></td>
</tr>
</tbody>
</table>
;TRANSFER: This section controls the data transfer from the FIFO to the printer.

OOAB 20A2FD
OOAB C2A4
00AD F2
00AE D2A4
00B0 C2A5
00B2 30B4FD
00B5 20B4FD
00B8 F2
00B9 D2A5
00BB 20A2EA
00BE C2A4
00C0 30B4FD
00C3 20B4FD
00C6 F2
00CD 01AE

; CKFIFO: JB ORDY,
CLR LB50
"SHIFT-OUT LOWER BYTE"

NEXT: SETB LB50
"DISABLE LOWER BYTE"

LB50
"SHIFT-OUT UPPER BYTE"

OOB2 759011

SELGEN: MOV P1.011H
"OUTPUT SELECT REQUEST"

MOV X @R0, A
"GENERATE "DATA STROBE" (NR PULSE)"

MOV T1, OFFH
"LOAD TIMER 1 FOR 75mSEC"

MOV TH1, 0BH

SETB ET1
"ENABLE TIMER 1 INTERRUPT"

SETB EA
"ENABLE INTERRUPTS"

SETB TR1
"START TIMER 1"

JB SELECT,$
"WAIT FOR "SELECT"

CLR EA
"DISABLE INTERRUPTS"

CLR ET1
"DISABLE TIMEK"

CLR TR1
"STOP TIMER"

MOV P1.0, OFFH
"SET PORT 1 AS INPUT"

SETB EX1
"ENABLE INT1 INTERRUPT ("FAULT")

ASSEMBLY COMPLETE, NO ERRORS FOUND
## TABLE 2
### ASCII CONVERSION CHART

<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 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
       MOV A, R4 // Convert lower nibble
       ANL A, #0FH // Clear upper nibble
       ADD A, #90H // Add 90Hex to nibble
       DA A // Convert nibble
       ADDC A, #40H // Add 40Hex to adjusted byte
       DA A // Convert nibble
       MOV @R1, A // Save ASCII converted value
       INE R1 // Increment address
       MOV A, R4 // Convert upper nibble
       SWAP A // Convert upper nibble
       ANL A, #0FH // Clear upper nibble
       ADD A, #90H // Add 90Hex to nibble
       DA A // Convert nibble
       ADDC A, #40H // Add 40Hex to adjusted byte
       DA A // Convert nibble
       MOV @R1, A
       INC R1 // Jump if upper byte completed
       DJNZ R7, ADDCK // Shift out upper byte
       CLR UB50 // Save upper byte in R4
       MOV R4, P1
       SETB UB50
       INC R7 // Status = 1
       AJMP CONV // Convert upper byte

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 (INT0) 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:
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:

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

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:

\[
\text{Transfer Rate} = \frac{132 \text{ bytes}}{(20\mu\text{sec.}/\text{clock})(20\text{clocks/div.})(5.8 \text{ div.})} = 56.9 \text{ KBytes/sec.}
\]

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

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

Pulse Width

\[
= 40 \text{ millisecond.}
\]

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 (TFO). When overflow occurs, the counter is automatically reloaded to the contents of TH0, (FFHex). Therefore, by clearing TFO 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.)
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
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
CALL BYTE SAVE

CALL TRANSFER FROM RAM

WAIT FOR NEXT EXECUTE REQUEST
BYTE TRANSFER

FIFO EMPTY?

SHIFT OUT LOWER BYTE

GENERATE "DATA STROBE" USING MOVX

SHIFT OUT UPPER BYTE

"ACKNOWLEDGE" RECEIVED?

YES

GENERATE "DATA STROBE" USING MOVX

FIFO EMPTY?

RETURN

NO

NO

"ACKNOWLEDGE" RECEIVED?

YES

SHIFT OUT LOWER BYTE

NO
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?

YES

RETURN

NO

NO
TRANSFER
FROM RAM

SET STARTING
RAM ADDRESS
TO 30HEX

END
OF STORED
BYTES?

YES

NO

OUTPUT CONTENTS
OF RAM TO PRINTER

GENERATE
"DATA STROBE"

INCREMENT
RAM ADDRESS

"ACKNOWLEDGE"
RECEIVED?

YES

NO

DECREMENT
REPETITION #

LAST
RAM
LOCATION?

YES

NO

END
REPETITION?

RETURN
YES

NIBBLE SAVE

SET STARTING RAM ADDRESS TO 30HEX

YES

FIFO EMPTY ?

NO

SHIFT OUT LOWER BYTE

CONVERT LOWER NIBBLE AND SAVE IN RAM

INCREMENT RAM ADDRESS

RETURN

NO

INCREMENT RAM ADDRESS

CONVERT UPPER NIBBLE AND SAVE IN RAM

INCREMENT RAM ADDRESS

CONVERT LOWER NIBBLE AND SAVE IN RAM

INCREMENT RAM ADDRESS

SHIFT OUT UPPER BYTE

YES

RAM FULL ?

INCREMENT RAM ADDRESS

CONVERT UPPER NIBBLE AND SAVE IN RAM

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 (Ø)

SAVE CHARACTER IN RAM

INCREMENT RAM ADDRESS

CHARACTER = "Z" ?

YES

RETURN

NO

INCREMENT TO NEXT CHARACTER
INPUT PIN DECLARATIONS

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>00A0</td>
<td></td>
<td>18</td>
<td>BUSY   BIT P2.0</td>
</tr>
<tr>
<td>00A1</td>
<td></td>
<td>19</td>
<td>PE     BIT P2.1</td>
</tr>
<tr>
<td>00A2</td>
<td></td>
<td>20</td>
<td>ORDY   BIT P2.2</td>
</tr>
<tr>
<td>00A3</td>
<td></td>
<td>21</td>
<td>UBDK   BIT P2.3</td>
</tr>
<tr>
<td>00B1</td>
<td></td>
<td>22</td>
<td>SELECT BIT P3.1</td>
</tr>
<tr>
<td>00B2</td>
<td></td>
<td>23</td>
<td>EX     BIT P3.2</td>
</tr>
<tr>
<td>00B3</td>
<td></td>
<td>24</td>
<td>FAULT  BIT P3.3</td>
</tr>
<tr>
<td>00B4</td>
<td></td>
<td>25</td>
<td>ACKNLG BIT P3.4</td>
</tr>
</tbody>
</table>

OUTPUT PIN DECLARATIONS

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>00A4</td>
<td></td>
<td>30</td>
<td>LBSO   BIT P2.4</td>
</tr>
<tr>
<td>00A5</td>
<td></td>
<td>31</td>
<td>UBSO   BIT P2.5</td>
</tr>
<tr>
<td>00B0</td>
<td></td>
<td>32</td>
<td>INPR1  BIT P3.0</td>
</tr>
<tr>
<td>00B6</td>
<td></td>
<td>33</td>
<td>STRB   BIT P3.6</td>
</tr>
</tbody>
</table>

36+1 $EJECT
LOC OBJ LINE SOURCE

37 ;INTERRUPT SERVICE ROUTINES
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83

ORG 00H
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 OBH ;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 1BH ;75mSEC TIMER ROUTINE
CLR EA ;DISABLE INTERRUPTS
CLR ET1 ;DISABLE TIMER 1 INTERRUPT
CLR TR1 ;DISABLE TIMER 1
AJMP CONT ;CONTINUE

+1 $EJECT
;***************************************************************************
84 ; ****************************
85 ; ****************************
86 ; ****************************
87 ; ****************************
88 ; ****************************
89 ; ****************************
90 ; ****************************
91 ;
92 ;
93 ORG 43H
94 INIT: MOV A,#0FFH ;SET UP I/O PORTS
95 MOV P1,A
96 MOV P2,A
97 MOV P3,A
98 MOV TMOHi,#11H ;TIMERS 0 & 1 SET AT 16-BITS
99 CLR INPRI ;SET "INPUT PRIME"
100 MOV TLOHi,#0CH ;WITH 40mSEC IN TIMER 0
101 MOV TH0Hi,#07H ;
102 SETB ETO ;ENABLE TIMER 0 INTERRUPT
103 SETB EA ;ENABLE INTERRUPTS
104 SETB TRO ;START TIMER 0
105 SJMP $ ;WAIT FOR TIMEOUT
106 EXEC: SETB INPRI ;REMOVE "INPUT PRIME"
107 MOV SP,#07H ;RESET STACK POINTER
108 MOV DPTR,#RET1 ;SET UP NEW INTERRUPT RETURN
109 PUSH DPL
110 PUSH DPH
111 RETI ;RETURN FROM INTERRUPT
112 RET1: SETB EX0 ;ENABLE EXECUTE INTERRUPT
113 SETB EA ;ENABLE INTERRUPTS
114 SJMP $ ;WAIT FOR EXECUTE REQUEST
115 CONT: MOV SP,#07H ;RESET STACK POINTER
116 MOV DPTR,#RET2 ;SET UP NEW INTERRUPT RETURN
117 PUSH DPL
118 PUSH DPH
119 RETI
120 RET2: ACALL SELGEN ;GENERATE "SELECT"
121 SETB EA ;ENABLE INTERRUPTS
122 AJMP CMD ;GO TO COMMAND DECODE
123 ;
124 ;
125 +1 $EJECT
SOURCE: This routine removes one 16-bit word from the FIFO, decodes the lower byte for desired command (01Hex- Byte Transfer, 02Hex- Nibble Transfer, or 03Hex- Test Generation), then the upper byte contains the number of repetitions for transfers.

0082 20A2FD
0085 C2A4
0087 AA90 0089 D2A4
008B C2A5
008D AB90
008F D2A5
0091 BA010D
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

CMD: JB ORU,$; WAIT FOR FIFO READY
CLR LBSO; SHIFT OUT LOWER BYTE
MOV R2,P1; SAVE LOWER BYTE IN R2
SETB LBSO
CLR UBSO; SHIFT OUT UPPER BYTE
MOV R3,P1; SAVE UPPER BYTE IN R3
SETB UBSO

0091 BA010D
0094 BB0004
0097 3166
0099 01B5
009B 314C
009D 5100
009F 01B5

CJNE R2,$001,NIBCK; JMP IF NOT NIBBLE TRANSFER
CJNE R3,$000,BREP; JMP IF REPEATED TRANSFER
ACALL BYTE; CALL BYTE TRANSFER SUBROUTINE
AJMP STOP

NIBCK: CJNE R2,$002;TESTCK; JMP IF NOT NIBBLE TRANSFER
ACALL NIB; CALL NIBBLE TRANSFER SUBROUTINE
ACALL TRAM; CALL TRANSFER RAM SUBROUTINE
AJMP STOP

TESTCK: CJNE R2,$003;NOCMD; JMP IN NOT TEST REQUEST
ACALL TEST; CALL TEST GENERATION SUBROUTINE
ACALL TRAM; CALL TRANSFER RAM SUBROUTINE
AJMP STOP

NOCMD: ACALL BYTE; IF INVALID COMMAND, TRANSFER BYTES

STOP: AJMP RET1; GO WAIT FOR NEXT COMMAND

16B +1 $EJECT
; CHECK FIFO STATUS FOR "FAULT" INTERRUPT

INTRET: POP DPH          ; REMOVE ADDRESS WHERE INTERRUPT OCCURRED
        POP DPL          ; LOWER BYTE OF ADDRESS IN DPL
        MOV R4,DPL      ; SAVE LOWER BYTE IN R4
        MOV R5,DPH      ; SAVE UPPER BYTE IN R5
        MOV DPTR,#RET3  ; SET UP NEW INTERRUPT RETURN
        PUSH DPL
        PUSH DPH
        RET

RET3: CJNE R5,#O2H,DI  ; JUMP IF INTERRUPTED IN BYTE SUB.
        AJMP IRAM       ; GO TO RAM SUBROUTINE

BI: MOV DPTR,#NEXT     ; FIND WHERE INTERRUPT OCCURRED
        MOV A,R4
        CLR C
        SUBB A,R4        ; COMPARE "NEXT" ADD. TO INTERRUPT ADD.
        JNC BYTE         ; JUMP IF INTERRUPT ADD. =< "NEXT"
        MOV DPTR,#CHECK
        MOV A,R4
        CLR C
        SUBB A,R4        ; COMPARE "CHECK" ADD. TO INTERRUPT ADD.
        JC BYTE          ; JUMP IF INTERRUPT ADD. =< "CHECK"
        CLR UBSO         ; SHIFT OUT UPPER BYTE
        SETB EA          ; ENABLE INTERRUPTS
        AJMP UPPER       ; SEND UPPER BYTE

; Selen: This subroutine "selects" the printer
; AND then enables the "fault" interrupt.

Selen: MOV P1,#11H     ; OUTPUT SELECT REQUEST
        MOVX @RO,A     ; GENERATE "DATA STROBE"
        MOV TL1,#OFFH ; LOAD TIMER 1 FOR 75mSEC
        MOV TH1,#0BH  ;
        SETB E1        ; ENABLE TIMER 1 INTERRUPT
        SETB EA         ; ENABLE INTERRUPTS
        SETB TR1        ; START TIMER 1
        JB SELECT,$     ; WAIT FOR "SELECT"
        CLR EA          ; DISABLE INTERRUPTS
        CLR E1          ; DISABLE TIMER
        CLR TR1          ; STOP TIMER
        MOV P1,#0FFH    ; SET PORT 1 AS INPUT
        SETB EX1         ; ENABLE INT1 INTERRUPT ("FAULT")
        RET

218 $EJECT
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 TRANSFER: This subroutine transfers bytes directly from the FIFO to the printer until the FIFO is empty.

BYTE: SETB EA ;Enable interrupts
      JB ORDY,END ;Wait for FIFO ready
      CLR 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 ACKNLG, ;Wait for "acknowledge"
             JB ACKNLG, ;
             MO VX @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 ACKNLG, ;Wait for "acknowledge"
                      JB ACKNLG, ;
                      MOVX @RO,A ;"data strobe" for lower byte
                      A JMP NEXT ;Send next byte
              BEND: CLR EA ;Disable interrupts
                      RET

BSAVE: MOV R1,#30H ;Set starting RAM address to 30Hex
       JB ORDY,EMPTY ;Jump if FIFO empty
BYTE: CLR LSB0 ;Shift out lower byte
       MO VX @RO,P1 ;Save lower byte in internal RAM
       INC R1 ;Increment address
       CLR USB0 ;Shift out upper byte
       MO VX @RO,P1 ;Save upper byte in internal RAM
       INC R1 ;Increment address
       JB ORDY1BEND ;Jump if FIFO empty
       CJNE R1,#BOH,BYTE ;Jump if address < B0Hex

EMPTY: RET

014C 7930
014E 20A214
0151 C2A4
0153 A790
0155 D2A4
0157 09
0158 C2A5
015A A790
015C D2A5
015E 09
015F 20A203
0162 980EC
0165 22
0166 D2AF
0168 20A21E
016B C2A4
016D F2
016E D2A4
0170 C2A5
0172 30B4FD
0175 20B4FD
0178 F2
0179 D2A5
017B 20A2EB
017E C2A4
0180 30B4FD
0183 20B4FD
0186 F2
0187 216E
0189 C2AF
018B 22
270 +1 $EJECT
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>271</td>
<td></td>
<td></td>
<td>;************ 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).</td>
</tr>
<tr>
<td>272</td>
<td></td>
<td></td>
<td>;************</td>
</tr>
<tr>
<td>273</td>
<td></td>
<td></td>
<td>;************</td>
</tr>
<tr>
<td>274</td>
<td></td>
<td></td>
<td>;************</td>
</tr>
<tr>
<td>275</td>
<td></td>
<td></td>
<td>;************</td>
</tr>
<tr>
<td>276</td>
<td></td>
<td></td>
<td>;************</td>
</tr>
<tr>
<td>277</td>
<td></td>
<td></td>
<td>;************</td>
</tr>
<tr>
<td>278</td>
<td></td>
<td></td>
<td>;************</td>
</tr>
<tr>
<td>279</td>
<td></td>
<td></td>
<td>;************</td>
</tr>
<tr>
<td>280</td>
<td></td>
<td></td>
<td>;************</td>
</tr>
<tr>
<td>281</td>
<td></td>
<td></td>
<td>;************</td>
</tr>
<tr>
<td>282</td>
<td></td>
<td></td>
<td>;************</td>
</tr>
<tr>
<td>283</td>
<td></td>
<td></td>
<td>;************</td>
</tr>
<tr>
<td>284</td>
<td></td>
<td></td>
<td>;************</td>
</tr>
</tbody>
</table>

```assembly
018C 7F01
018E 7930
0190 204220
0193 1F
0194 C2A4
0196 AC90
0198 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
01A9 2490
01AC D4
01AD 3440
01AF D4
01B0 F7
01B1 09
01B2 DF09
01B4 C2A5
01B6 AC90
01BB D2A5
01BA 0F
01BB 219A
01BD B980B0
01C0 22
```

```
0185 MOV R7, #01H ; UPPER OR LOWER BYTE STATUS FLAG
0186 MOV R1, #30H ; INITIALIZE ADDRESS LOCATION
0187 NBX: JB ORDY/DONE ; JUMP IF FIFO EMPTY
0188 DEC R7 ; STATUS = 0
0189 CLR LB50 ; SHIFT OUT LOWER BYTE
018A MOV R4, P1 ; SAVE LOWER BYTE IN R4
018B SETB LB50
018C CONV: INC R7 ; STATUS INCREMENTED
018D MOV A, R4
018E ANL A, #0FH ; CLEAR UPPER NIBBLE
018F ADD A, #90H ; ADD 90HEX TO NIBBLE
0190 DA A
0191 ADDC A, #40H ; ADD 40HEX TO ADJUSTED BYTE
0192 DA A
0193 MOV @R1, A ; SAVE ASCII CONVERTED VALUE
0194 INC R1 ; INCREMENT ADDRESS
0195 MOV A, R4
0196 SWAP A ; CONVERT UPPER NIBBLE
0197 ANL A, #0FH
0198 ADD A, #70H
0199 DA A
019A ADDC A, #40H
019B DA A
019C MOV @R1, A
019D INC R1
019E DJNZ R7, ADDCK ; JUMP IF UPPER BYTE COMPLETED
019F CLR UB50 ; SHIFT OUT UPPER BYTE
01A0 MOV R4, P1 ; SAVE UPPER BYTE IN R4
01A1 SETB UB50
01A2 INC R7 ; STATUS = 1
01A3 DJNZ R7, ADDCK ; JUMP IF ADDRESS < BOHEX
01A4 ADDCK: CJNE R1, #00H, NXTNB ; JUMP IF ADDRESS < 00HEX
01A5 DONE: RET
01A6 319 +1 $EJECT
```
ORG 200H

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.

ENTRY VALUES:
R1 = Address of terminating location,
R3 = Number of repetitions
RAH

DATA IS TO BE
TRANSFERRED.

TRAH:
REPT:
NEXTB:
INTB:
IREP:
ENDBS:
SAVE
TERMINATING ADDRESS
IN R7
ENABLE INTERRUPTS
SETUP STARTING ADDRESS
INC R3
XRL A1R7
JZ ENDBS
PARCE TERMINATING ADD. TO START ADD. IF SAME, JUMP TO ENDBS
MOV R1,#10
MOVX A1R7
INC R1
XRL A1R7
JNZ NEXTB
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
COMPARE CURRENT ADD. TO TERMINATE ADD.
TEST GENERATION: This subroutine generates 59 characters to be printed (20Hex to 5AHex) and saves them in internal RAM (locations 30 Hex to 6AHex).

TEST: MOV R1,#30H ;SET STARTING ADDRESS
MOV R5,#20H ;STARTING CHARACTER=20HEX( )
MOV R6,#5AH ;ENDING CHARACTER=5AH( )
NXTST: MOV A,R5 ;SAVE CHARACTER AT [R1]
MOV @R1,A
XRL A,R6 ;COMPARE LAST CHARACTER TO "Z"
JZ TEND ;JUMP IF "Z"
INC R1 ;INCREMENT CHARACTER
INC R5 ;GO TO NEXT CHARACTER
AJMP NXTST ;SAVE NEXT CHARACTER
TEND: MOV R1,#6BH ;PLACE TERMINATING ADDRESS INTO R1
RET ;
END

ASSEMBLY COMPLETE, NO ERRORS FOUND
BYTE TRANSFER ROUTINE FOR C8751-8 (5-8 MHz)
TRANSFER RAM: This subroutine transfers data from internal RAM (locations 30HEX to CR13) to the printer.

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

\[\text{TRAM: MOV A,R1} \] ; SAVE TERMINATING ADDRESS
\[\text{MOV R7,A} \] ; IN R7
\[\text{INC R3} \] ; SET INDEX TO 8-BIT COUNTER
\[\text{MOV TMOD,#16H} \] ; SET MODulo as 8-BIT COUNTER
\[\text{MOV TL0,#0FFH} \] ; SET LOW BYTE OF COUNTER TO FF
\[\text{MOV TH0,#0FFH} \] ; SET RELOAD VALUE TO FF
\[\text{CLR TF0} \] ; CLEAR OVERFLOW Bit
\[\text{SETB TR0} \] ; ENABLE COUNTER
\[\text{SETB EA} \] ; ENABLE INTERRUPTS
\[\text{MOV R1,#01H} \] ; SET UP STARTING ADDRESS
\[\text{MOV A,R1} \] ;
\[\text{XRL A,R7} \] ; COMPARE TERMINATING ADDR. TO START ADDR.
\[\text{JZ ENDBS} \] ; IF SAME, JUMP TO ENDBS
\[\text{REPT: MOV P1,R1} \] ; OUTPUT RAM CONTENTS TO PRINTER
\[\text{NEXTB: MOVX @R0,A} \] ; GENERATE "DATA STROBE" (WR)
\[\text{INC R1} \] ; INCREMENT ADDRESS
\[\text{MOV P1,R1} \] ; OUTPUT NEXT RAM CONTENTS TO PRINTER
\[\text{MOV A,R1} \] ;
\[\text{JNZ NEXTB} \] ; IF NOT EQUAL, OUTPUT NEXT DATA
\[\text{JNB TF0,0} \] ; IF NOT equal, output next DATA
\[\text{DECR R3} \] ; DECREMENT REPETITION #
\[\text{IF = 0 GO TO ENDBS} \] ;
\[\text{CJNE R3,#00H,REPT} \] ; IF NOT EQUAL, NO REPEAT TRANSFER
\[\text{AINP ENDBS} \] ; IF = 0 GO TO ENDBS
\[\text{IRAN: SETB EA} \] ; ENABLE INTERRUPTS
\[\text{MOV A,R1} \] ; LOAD ADDRESS INTO ACC.
\[\text{AJMP INTR} \] ;
\[\text{ENDBS: CLR EA} \] ; DISABLE INTERRUPTS
\[\text{RET} \] ;

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

FIFO SPECIFICATION
First-In First-Out (FIFO) 64X4 Serial Memory

Monolithic Memories 67401 (11)

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**

- IR - Input Ready
- SI - Shift In
- MR - Master Reset
- OR - Output Ready
- SO - Shift Out
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, indication 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>f₁</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>f₀</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>
</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** - (ACKNOWLEDGE) 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.
POWER-UP
CLEAR

SELECT

BUSY

DATA STROBE

ACKNLG

DATA

HEX 11

A = 1.0 microsec. min.
B = 0.5 microsec. min.
1.0 microsec. max.

C = 4+ 1.0 microsec.
D = Greater than zero

SELECT GENERATION VIA DATA BUS
SELECT

DATA STROBE

ACKNLG

DATA

BUSY

DATA TRANSFER TIMING WITHOUT BUSY

A = 1.0 microsec.min.

B = 0.5 microsec.min.

C = 7+1.2 microsec.

D = 4+1.0 microsec.

E = Greater than zero

F = 0.0 microsec.min.

G = 0.075 microsec.max.
**SELECT**  
**DATA STROBE**  
**BUSY**  
**ACKNLG**  
**DATA**

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

**DATA TRANSFER TIMING WITH BUSY**
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.