An Evaluation of the 8051 Microcontroller

Spring 1983

George C. Schafner
University of Central Florida

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

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

Part of the Engineering Commons

STARS Citation

https://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 lee.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,#8ACH</td>
<td>MOV P1,#8ACH</td>
</tr>
<tr>
<td>OUTL P1,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.
IIII. 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
IR
SI
64X8
FIFO
OR SO

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>64th Word</th>
<th>Upper Byte</th>
<th>Lower Byte</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Last Byte</td>
<td>127th</td>
</tr>
<tr>
<td></td>
<td>To Be</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Printed</td>
<td></td>
</tr>
<tr>
<td>2nd Word</td>
<td>4th</td>
<td>5th</td>
</tr>
<tr>
<td>1st Word</td>
<td>2nd</td>
<td>1st Byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To Be</td>
</tr>
<tr>
<td></td>
<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 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)
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 INT1 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)
Figure 8. Flowchart of logical control sequence
sequence is modified for reduced delay as shown by the flow-chart 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 flow-chart 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
OUTPUT "SELECT" REQUEST (11HEX) TO PORT 1

GENERATE "DATA STROBE" PULSE USING MOVX INSTRUCTION

START 75 MSEC FAILSAFE TIMER

"SELECT" RECEIVED?

NO

DISABLE TIMER

SET PORT 1 AS AN INPUT

ENABLE INTERRUPT FOR "FAULT" OCCURENCE

RETURN

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 ACKNL6 BIT P3.4 ; ACKNOWLEDGE FROM PRINTER (HIGH TRUE)

OUTPUT PIN DECLARATIONS

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)
4 +1 $EJECT
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td></td>
<td></td>
<td>;*****************************************</td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
<td>;*****************************************</td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td>;*****                           ****</td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td>;*****                       INTERRUPT SERVICE ROUTINES         ****</td>
</tr>
<tr>
<td>41</td>
<td></td>
<td></td>
<td>;*****                           ****</td>
</tr>
<tr>
<td>42</td>
<td></td>
<td></td>
<td>;*****************************************</td>
</tr>
<tr>
<td>43</td>
<td></td>
<td></td>
<td>;*****************************************</td>
</tr>
<tr>
<td>44</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>45</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>46</td>
<td></td>
<td></td>
<td>ORG OOH    ; GO TO INIT AT RESET</td>
</tr>
<tr>
<td>47</td>
<td></td>
<td></td>
<td>AJMP INIT    ; GO TO INIT AT RESET</td>
</tr>
<tr>
<td>48</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>49</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td>;*****************************************</td>
</tr>
<tr>
<td>51</td>
<td></td>
<td></td>
<td>;*****************************************</td>
</tr>
<tr>
<td>52</td>
<td></td>
<td></td>
<td>ORG 03H    ; EXECUTE INTERRUPT ROUTINE</td>
</tr>
<tr>
<td>53</td>
<td></td>
<td></td>
<td>CLR EA    ; DISABLE INTERRUPTS</td>
</tr>
<tr>
<td>54</td>
<td></td>
<td></td>
<td>CLR EXO    ; DISABLE EXTERNAL INTERRUPT 0</td>
</tr>
<tr>
<td>55</td>
<td></td>
<td></td>
<td>AJMP CONT    ; CONTINUE</td>
</tr>
<tr>
<td>56</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>57</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>58</td>
<td></td>
<td></td>
<td>;*****************************************</td>
</tr>
<tr>
<td>59</td>
<td></td>
<td></td>
<td>;*****************************************</td>
</tr>
<tr>
<td>60</td>
<td></td>
<td></td>
<td>ORG 0BH    ; 40mSEC TIMER ROUTINE</td>
</tr>
<tr>
<td>61</td>
<td></td>
<td></td>
<td>CLR EA    ; DISABLE INTERRUPTS</td>
</tr>
<tr>
<td>62</td>
<td></td>
<td></td>
<td>CLR ET0    ; DISABLE TIMER 0 INTERRUPT</td>
</tr>
<tr>
<td>63</td>
<td></td>
<td></td>
<td>CLR TR0    ; DISABLE TIMER 0</td>
</tr>
<tr>
<td>64</td>
<td></td>
<td></td>
<td>AJMP EXEC    ; WAIT FOR EXECUTE REQUEST</td>
</tr>
<tr>
<td>65</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>66</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>67</td>
<td></td>
<td></td>
<td>;*****************************************</td>
</tr>
<tr>
<td>68</td>
<td></td>
<td></td>
<td>;*****************************************</td>
</tr>
<tr>
<td>69</td>
<td></td>
<td></td>
<td>ORG 13H    ; &quot;FAULT&quot; INTERRUPT ROUTINE</td>
</tr>
<tr>
<td>70</td>
<td></td>
<td></td>
<td>CLR EA    ; DISABLE INTERRUPTS</td>
</tr>
<tr>
<td>71</td>
<td></td>
<td></td>
<td>AJMP INTRET    ; GO TO INTRET</td>
</tr>
<tr>
<td>72</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>73</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>74</td>
<td></td>
<td></td>
<td>;*****************************************</td>
</tr>
<tr>
<td>75</td>
<td></td>
<td></td>
<td>;*****************************************</td>
</tr>
<tr>
<td>76</td>
<td></td>
<td></td>
<td>ORG 18H    ; 75mSEC TIMER ROUTINE</td>
</tr>
<tr>
<td>77</td>
<td></td>
<td></td>
<td>CLR EA    ; DISABLE INTERRUPTS</td>
</tr>
<tr>
<td>78</td>
<td></td>
<td></td>
<td>CLR ET1    ; DISABLE TIMER 1 INTERRUPT</td>
</tr>
<tr>
<td>79</td>
<td></td>
<td></td>
<td>CLR TR1    ; DISABLE TIMER 1</td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
<td>AJMP CONT    ; CONTINUE</td>
</tr>
<tr>
<td>81</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>82</td>
<td></td>
<td></td>
<td>;</td>
</tr>
<tr>
<td>83</td>
<td>+1</td>
<td>$EJECT</td>
<td></td>
</tr>
</tbody>
</table>
INITIALIZE: This routine sets up I/O Ports, awaits execution request, and resets the printer.

INIT: ORG $43H ;SET UP I/O PORTS

MOV A,$OFFH
MOV P1A
MOV P2A
MOV P3A
MOV TMOD,$11H ;TIMERS 0 & 1 SET AT 16-BITS
CLR INPRI ;SET "INPUT PRIME"
MOV TLO,$0C9H ;INIT 40mSEC IN TIMER 0
MOV TH0,$07DH

SETB ETO ;ENABLE TIMER 0 INTERRUPT
SETB EA ;ENABLE INTERRUPTS
SETB TR0 ;START TIMER 0
SJMP ; ;WAIT FOR TIMEOUT

EXEC: SETB INPRI ;REMOVE "INPUT PRIME"
MOV SP,$07H ;RESET STACK POINTER
MOV DPTR,#RET1 ;SET UP NEW INTERRUPT RETURN

PUSH DPL
PUSH DPH

RETI ;RETURN FROM INTERRUPT

RET1: SETB EX0 ;ENABLE EXECUTE INTERRUPT
SETB EA ;ENABLE INTERRUPTS
SJMP ; ;WAIT FOR EXECUTE REQUEST

CONT: MOV SP,$07H ;RESET STACK POINTER
MOV DPTR,#RET2 ;SET UP NEW INTERRUPT RETURN

PUSH DPL
PUSH DPH

RETI

RET2: ACALL SELGEN ;GENERATE "SELECT"

SJMP ; ;EXIT

A JMP CKFIFO ;START TRANSFER

+1 $EJECT
### LOC OBJ | LINE | SOURCE
--- | --- | ---
124 | ; | CHECK FIFO STATUS FOR "FAULT" INTERRUPT
125 | ;

```
0082 D0B3 126 INTRET: POP DPH \#REMOVE ADDRESS WHERE INTERRUPT OCCURRED
0084 D0B2 127 POP DPL \#LOWER BYTE OF ADDRESS IN DPL
0086 ACB2 128 MOV R4,DPL \#SAVE LOWER BYTE IN R4
0088 90090 129 MOV DPTR,#RET3 \#SET UP NEW INTERRUPT RETURN
008B C0B2 130 PUSH DPL
008D C0B3 131 PUSH DPH
008F 32 132 RETI

0090 900AE 133 RET3: MOV DPTR,#NEXT \#FIND WHERE INTERRUPT OCCURED

0093 E5B2 134 MOV A,DPL
0095 C3 135 CLR C
0096 9C 136 SUBB A,R4 \#COMPARE "NEXT" ADDR. TO INTERRUPT ADDR.
0097 500F 137 JNC CKFIFO \#JUMP IF INTERRUPT ADDR. =< "NEXT"
0099 900EB 138 MOV DPTR,#CHECK
009C E5B2 139 MOV A,DPL
009E C3 140 CLR C
009F 9C 141 SUBB A,R4 \#COMPARE "CHECK" ADDR. TO INTERRUPT ADDR.
00A0 4006 142 JC CKFIFO \#JUMP IF INTERRUPT ADDR. > "CHECK"
00A2 C2A5 143 CLR UBSG \#SHIFT OUT UPPER BYTE
00A4 D2AF 144 SETB EA \#ENABLE INTERRUPTS
00A6 01B8 145 AJMP UPPER \#SEND UPPER BYTE

146 ;
147 ;
148 +1 $EJECT
```
TRANSFER: This section controls the transfer from the FIFO to the printer.

; CB FIFO: JB ORDY,$ \text{ \textit{wait for FIFO ready}}
CLR LBSO \text{ \textit{shift-out lower byte}}
MOGX @RO,A \text{ \textit{generate "data strobe" (NR pulse)}}

NEXT: SETB LBSO \text{ \textit{disable lower byte}}
CLR UBSO \text{ \textit{shift-out upper byte}}

JNB ACKNLG,$ \text{ \textit{wait for "acknowledge"}}

JNB ACKNLG,$

UPPER: MOX @RO,A \text{ \textit{"data strobe" for upper byte}}

CHECK: SETB UBSO \text{ \textit{disable upper byte}}
JB ORDY,CKFIFO \text{ \textit{if FIFO not ready go to CKFIFO}}

CLR LBSO \text{ \textit{shift-out lower byte}}

JNB ACKNLG,$ \text{ \textit{wait for "acknowledge"}}

JNB ACKNLG,$

MOVX @RO,A \text{ \textit{"data strobe" for lower byte}}

AJMP NEXT \text{ \textit{send next byte}}

; \text{ \textit{select printer and enable fault interrupt}}

MOV P1,$11H \text{ \textit{output select request}}
MOX @RO,A \text{ \textit{generate "data strobe"}}
MOV TL1,OFFH \text{ \textit{load timer 1 for 75msec}}
MOV TH1,0BH

SETB ET1 \text{ \textit{enable timer 1 interrupt}}
SETB EA \text{ \textit{enable interrupts}}
SETB TR1 \text{ \textit{start timer 1}}

JB SELECT,$ \text{ \textit{wait for "select"}}
CLR EA \text{ \textit{disable interrupts}}
CLR ET1 \text{ \textit{disable timer}}
CLR TR1 \text{ \textit{stop timer}}

MOV P1,0FFH \text{ \textit{set port 1 as input}}
SETB EX1 \text{ \textit{enable INT1 interrupt ("fault")}}

RET

END
<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
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  ; CLEAR UPPER NIBBLE
       ANL A,#0FH  ; ADD 0FH TO NIBBLE
       ADD A,#90H  ; ADD 40H TO ADJUSTED BYTE
       ADDC A,#40H  ; SAVE ASCII CONVERTED VALUE
       DA A  ; INCREMENT ADDRESS
       MOV @kl1A  ; CONVERT UPPER NIBBLE
       INC R1  ; STATUS=1
       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
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 OllH 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:

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

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:

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

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 (TF0). When overflow occurs, the counter is automatically reloaded to the contents of TH0, (FFHex). Therefore, by clearing TF0 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" \(>1\mu\text{sec.}\)).
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
INITIALIZE

PULSE "INPUT PRIME" TO RESET PRINTER

EXECUTE REQUEST?

NO

CALL SELGEN ("SELECT" GENERATION)

YES

GO TO COMMAND DECODE
COMMAND
DECODE

FIFO
OUTPUT
READY
?

YES

SHIFT OUT
LOWER BYTE
AND SAVE IN R2

SHIFT OUT
UPPER BYTE
AND SAVE IN R3

BYTE
TRANSFER
COMMAND
?

YES

ZERO
REPETITION
?

YES

CALL BYTE
TRANSFER

NO

CALL BYTE
SAVE

CALL TRANSFER
FROM RAM

WAIT FOR NEXT
EXECUTE REQUEST

47
BYTE TRANSFER

FIFO EMPTY?

NO

SHIFT OUT LOWER BYTE

GENERATE "DATA STROBE" USING MOVX

SHIFT OUT UPPER BYTE

"ACKNOWLEDGE" RECEIVED?

NO

YES

GENERATE "DATA STROBE" USING MOVX

FIFO EMPTY?

NO

YES

RETURN

"ACKNOWLEDGE" RECEIVED?

NO

SHIFT OUT LOWER BYTE
BYTE SAVE

SET STARTING
RAM ADDRESS
TO 30HEX

FIFO
EMPTY?

YES

NO

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
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 REPLICATION?

NO

DECREMENT REPLICATION #
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

CONVERT UPPER NIBBLE AND SAVE IN RAM

INCREMENT RAM ADDRESS

RETURN

NO

INCREMEKT RAM ADDRESS

CONVERT UPPER NIBBLE AND SAVE IN RAM

INCREMENT RAM ADDRESS

CONVERT LOWER NIBBLE AND SAVE IN RAM

SHIFT OUT UPPER BYTE

RAM FULL?

YES

NO

RETURN
TEST
GENERATION

SET STARTING
RAM ADDRESS
TO 30HEX

SET STARTING
CHARACTER
TO 20 HEX (Ω)

SAVE CHARACTER
IN RAM

INCREMENT
RAM ADDRESS

CHARACTER
="Z"
?

NO
INCREMENT
TO NEXT
CHARACTER

YES
RETURN
INPUT PIN DECLARATIONS

00A0 18 BUSY BIT P2.0 \BUSY FROM PRINTER (LOW TRUE)
00A1 19 PE BIT P2.1 \PAPER EMPTY (LOW TRUE)
00A2 20 ORDY BIT P2.2 \FIFO OUTPUT READY (LOW TRUE)
00A3 21 UBOR BIT P2.3 \FIFO UPPER BYTE OUTPUT READY (LOW TRUE)
00B1 22 SELECT BIT P3.1 \SELECT FROM PRINTER (LOW TRUE)
00B2 23 EX BIT P3.2 \EXECUTE REQUEST (LOW TRUE)
00B3 24 FAULT BIT P3.3 \FAULT FROM PRINTER (LOW TRUE)
00B4 25 ACKNLG BIT P3.4 \ACKNOWLEDGE FROM PRINTER (HIGH TRUE)

OUTPUT PIN DECLARATIONS

00A4 30 LBSO BIT P2.4 \LOWER BYTE SHIFT OUT (LOW TRUE)
00A5 31 UBSD BIT P2.5 \UPPER BYTE SHIFT OUT (LOW TRUE)
00B0 32 INPPR BIT P3.0 \INPUT PRIME TO PRINTER (LOW TRUE)
00B6 33 STRB BIT P3.6 \DATA STROBE TO PRINTER (LOW TRUE)

+1 $EJECT
**INTERRUPT SERVICE ROUTINES**

0000

; ORG 00H
AJMP INIT ; GO TO INIT AT RESET

0003

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

000B

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

0013

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

001D

; ORG 1BH ; 75mSEC TIMER ROUTINE
CLR EA ; DISABLE INTERRUPTS
CLR ET1 ; DISABLE TIMER 1 INTERRUPT
CLR TR1 ; DISABLE TIMER 1
AJMP CONT ; CONTINUE
LDC OBJ
LINE
SOURCE
84 ;INITIALISATION: This routine sets up I/O
85 ;Ports; awaits execution request; and resets
86 ;the printer.
87
88
89
90

; -------------------------------

0043 93 ORG 43H ;SET UP I/O PORTS
0043 74FF 94 INIT: MOV A,#0FFH
0045 F590 95 MOV P1,A
0047 F5A0 96 MOV P2,A
0049 F5B0 97 MOV P3,A
004B 758911 98 MOV TH0,#011H ;TIMERS 0 & 1 SET AT 16-BITS
004E C280 99 CLR INPRI ;SET "INPUT PRIME"
0050 758AC9 100 MOV TL0,#OC9H ;WITH 40mSEC IN TIMER 0
0053 758C7D 101 MOV TH0,#078H ;
0056 D2A9 102 SETB ETO ;ENABLE TIMER 0 INTERRUPT
0058 D2AF 103 SETB EA ;ENABLE INTERRUPTS
005A D28C 104 SETB TRO ;START TIMER 0
005C 80FE 105 SJMP $ ;WAIT FOR TIMEOUT
005E D2B0 106 EXEC: SETB INPRI ;REMOVE "INPUT PRIME"
0060 758107 107 MOV SP,#07H ;RESET STACK POINTER
0063 90006B 108 MOV DPTR,#RET1 ;SET UP NEW INTERRUPT RETURN
0066 C082 109 PUSH DPL
0068 C083 110 PUSH DPH
006A 32 111 RETI ;RETURN FROM INTERRUPT
006B D2AB 112 RET1: SETB EX0 ;ENABLE EXECUTE INTERRUPT
006D D2AF 113 SETB EA ;ENABLE INTERRUPTS
006F 80FE 114 SJMP $ ;WAIT FOR EXECUTE REQUEST
0071 758107 115 CONT: MOV SP,#07H ;RESET STACK POINTER
0074 90007C 116 MOV DPTR,#RET2 ;SET UP NEW INTERRUPT RETURN
0077 C082 117 PUSH DPL
0079 C083 118 PUSH DPH
007B 32 119 RETI
007C 312D 120 RET2: ACALL SELGEN ;GENERATE "SELECT"
007E D2AF 121 SETB EA ;ENABLE INTERRUPTS
0080 0182 122 AJMP CMD ;GO TO COMMAND DECODE
123 }
124 }
125 +1 $EJECT
SOURCE

CMD: JB ORUT, $ 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

CJNE R2, #01H, NIBCK JMP IF NOT BYTE TRANSFER
CJNE R3, #00H, BREP JMP IF REPEATED TRANSFER
ACALL BYTE CALL BYTE TRANSFER SUBROUTINE
AJMP STOP

BREP: ACALL BSAVE CALL BYTE SAVE SUBROUTINE
ACALL TRAH CALL TRANSFER RAM SUBROUTINE
AJMP STOP

NIBCK: CJNE R2, #02H, NIBCK JMP IF NOT NIBBLE TRANSFER
ACALL NIB CALL NIBBLE TRANSFER SUBROUTINE
ACALL TRAM CALL TRANSFER RAM SUBROUTINE
AJMP STOP

TESTCK: CJNE R2, #03H, 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

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

INTRET: POP DPH  ; REMOVE ADDRESS WHERE INTERRUPT OCCURED
       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
       RETI

RET3: CJNE R5,#02H,BT  ; JUMP IF INTERRUPTED IN BYTE SUB.
       AJMP IRAM  ; GO TO TRAM SUBROUTINE

BT: MOV DPTR,#NEXT  ; IFIND WHERE INTERRUPT OCCURED
       MOV A,R4
       CLR C
       JNC BYTE  ; JUMP IF INTERRUPT ADDR. =< "NEXT"
       SUBB A,R4  ; COMPARE "NEXT" ADDR. TO INTERRUPT ADDR.
       JNC BYTE  ;
       JC BYTE  ;"CHECK"
       CLR UB0  ; SHIFT OUT UPPER BYTE
       SETB EA  ; ENABLE INTERRUPTS
       AJMP UPPER  ; SEND UPPER BYTE

; SELGEN: THIS SUBROUTINE "SELECTS" THE PRINTER
; AND THEN ENABLES THE "FAULT" INTERRUPT.

SELGEN: MOV P1,#11H  ; OUTPUT SELECT REQUEST
       MOVX @R0,A  ; GENERATE "DATA STROBE"
       MOV TL1,#0FFH  ; 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

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/EMPTY; wait for FIFO ready
    CLR LSB0; shift-out lower byte
    MOVX @RO+R; generate "data strobe" (WR pulse)
    NEXT: SETB LSB0; disable lower byte
    CLR USB0; shift-out upper byte
    JNB ACKNLG1$; wait for "acknowledge"
    JB ACKNLG1$; 

UPPER: MOVX @RO+R; "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 ACKNLG1$; wait for "acknowledge"
    JB ACKNLG1$; 
    MOVX @RO+R; "data strobe" for lower byte
    AJMP NEXT; send next byte
BEND: CLR EA; disable interrupts
RETF

BYTE: MOV R1,#30H; set starting RAM address to 30Hex
    JB ORDY/EMPTY; jump if FIFO empty
    LSB0; shift-out lower byte
    MOV @R0+P1; save lower byte in internal RAM
    SETB LSB0; 
    INC R1; increment address
    CLR USB0; shift-out upper byte
    MOV @R0+P1; save upper byte in internal RAM
    SETB USB0; 
    INC R1; increment address
    JB ORDY/EMPTY; jump if FIFO empty
    CJNE R1, #BOH/LBYTE; jump if address < BOHex

BYTE: SETB EA; enable interrupts
    JB ORDY/BEND; wait for FIFO ready
    CLR LSB0; shift-out lower byte
    MOVX @RO+R; generate "data strobe" (WR pulse)
    NEXT: SETB LSB0; disable lower byte
    CLR USB0; shift-out upper byte
    JNB ACKNLG1$; wait for "acknowledge"
    JB ACKNLG1$; 

UPPER: MOVX @RO+R; "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 ACKNLG1$; wait for "acknowledge"
    JB ACKNLG1$; 
    MOVX @RO+R; "data strobe" for lower byte
    AJMP NEXT; send next byte
BEND: CLR EA; disable interrupts
RETF

BYTE: MOV R1,#30H; set starting RAM address to 30Hex
    JB ORDY/EMPTY; jump if FIFO empty
    LSB0; shift-out lower byte
    MOV @R0+P1; save lower byte in internal RAM
    SETB LSB0; 
    INC R1; increment address
    CLR USB0; shift-out upper byte
    MOV @R0+P1; save upper byte in internal RAM
    SETB USB0; 
    INC R1; increment address
    JB ORDY/EMPTY; jump if FIFO empty
    CJNE R1, #BOH/LBYTE; jump if address < BOHex

BYTE: SETB EA; enable interrupts
    JB ORDY/BEND; wait for FIFO ready
    CLR LSB0; shift-out lower byte
    MOVX @RO+R; generate "data strobe" (WR pulse)
    NEXT: SETB LSB0; disable lower byte
    CLR USB0; shift-out upper byte
    JNB ACKNLG1$; wait for "acknowledge"
    JB ACKNLG1$; 

UPPER: MOVX @RO+R; "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 ACKNLG1$; wait for "acknowledge"
    JB ACKNLG1$; 
    MOVX @RO+R; "data strobe" for lower byte
    AJMP NEXT; send next byte
BEND: CLR EA; disable interrupts
RETF

BYTE: MOV R1,#30H; set starting RAM address to 30Hex
    JB ORDY/EMPTY; jump if FIFO empty
    LSB0; shift-out lower byte
    MOV @R0+P1; save lower byte in internal RAM
    SETB LSB0; 
    INC R1; increment address
    CLR USB0; shift-out upper byte
    MOV @R0+P1; save upper byte in internal RAM
    SETB USB0; 
    INC R1; increment address
    JB ORDY/EMPTY; jump if FIFO empty
    CJNE R1, #BOH/LBYTE; jump if address < BOHex

BYTE: SETB EA; enable interrupts
    JB ORDY/BEND; wait for FIFO ready
    CLR LSB0; shift-out lower byte
    MOVX @RO+R; generate "data strobe" (WR pulse)
    NEXT: SETB LSB0; disable lower byte
    CLR USB0; shift-out upper byte
    JNB ACKNLG1$; wait for "acknowledge"
    JB ACKNLG1$; 

UPPER: MOVX @RO+R; "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 ACKNLG1$; wait for "acknowledge"
    JB ACKNLG1$; 
    MOVX @RO+R; "data strobe" for lower byte
    AJMP NEXT; send next byte
BEND: CLR EA; disable interrupts
RETF
;******************************************************************************
; 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).
;******************************************************************************

ENTRY:
; LOOP
271: MOV R7,#01H
272: MOV R1,#30H
273: JB ORY,DONE
274: DEC R7
275: JSR 0190 20H
276: CLR LB50
277: MOV R4,P1
278: JSR 019B 22H
279: JSR 019A 0FH
280: MOV A,R4
281: INC R7
282: JSR 019B 23H
283: JSR 019D 24H
284: 

NXTNib:
285: MOV R7,#01H
286: MOV R1,#30H
287: JB ORY,DONE
288: DEC R7
289: JSR 0190 20H
290: CLR LB50
291: MOV R4,P1
292: JSR 019B 22H
293: JSR 019A 0FH
294: MOV A,R4
295: INC R7
296: JSR 019B 23H
297: JSR 019D 24H

CONV:
298: MOV A,R4
299: ADD A,#90H
300: INC R1
301: MOV A,R4
302: SWAP A
303: MOV A,#0FH
304: ADD A,#90H
305: INC R1
306: ADD A,#40H
307: MOV A,#0FH
308: MOV A,#90H
309: MOV A,#40H

ADDCK:
310: DJNZ R7,ADDCK
311: CLR UD50
312: MOV R4,P1
313: SETB UD50
314: INC R7
315: AJMP CONV
316: ADDCK: CJNE R1,#80H,NXTNib
317: DONE: RET
318: 
319 +1 "$EJECT
LOC  OBJ  LINE  SOURCE
320       ORG 200H
321
322
323  ;****** TRANSFER RAM: THIS SUBROUTINE TRANSFERS DATA ******
324  ;****** FROM INTERNAL RAM (LOCATIONS 30HEX TO [R1]) ******
325  ;****** TO THE PRINTER.  ******
326  ;****** ******
327  ;****** ENTRY VALUES: R1 = ADDRESS OF TERMINATING ******
328  ;****** LOCATION.  ******
329  ;****** ******
330  ;****** R3 = NUMBER OF REPETITIONS  ******
331  ;****** RAM DATA IS TO BE  ******
332  ;****** TRANSFERRED.  ******
333  ;****** ******
334
335
336
0200 E9
337  TRAM: MOV A,R1  ;SAVE TERMINATING ADDRESS
338  MOV R7,A  ;IN R7
339
0201 FF
340
341  MOV R1,#30H  ;SET UP STARTING ADDRESS
342
0202 OB
343
344
0203 D2AF
345  REPT: MOV P1,#R1  ;OUTPUT RAM CONTENTS TO PRINTER
346  MOVX @R0,A  ;GENERATE "DATA STROBE" (WR)
347
0205 7930
348
349
0207 E9
350  XRL A,R7  ;COMPARE TERMINATING ADD. TO START ADD.
351
0208 6F
352  J2 ENDBS  ;IF SAME, JUMP TO ENDBS
353
0209 601D
354
355
020B B790
356  NEXTB: MOVX @R0,A  ;INCREMENT ADDRESS
357
020D F2
358
359
020E 09
360  MOV P1,#R1  ;OUTPUT NEXT RAM CONTENTS TO PRINTER
361
020F B790
362
363
0211 E9
364
365
0212 3084FD
366  JNB ACKNLG,R  ;WAIT FOR "ACKNOWLEDGE" FROM PRINTER
367
0215 2084FD
368
369
0218 6F
370  JB ACKNLG,R  ;ENABLE INTERRUPTS
371
0219 70F2
372
373
021B 7930
374  INTR: XRL A,R7  ;COMPARE CURRENT ADD. TO TERMINATE ADD
375
021D 1B
376  JNZ NEXTB  ;JUMP IF NOT EQUAL
377
021E B000EA
378
379
0221 412B
380  IAHR: MOV A,R1  ;LOAD ADDRESS INTO ACC.
381
0223 D2AF
382
383
0225 E9
384
385
0226 411B
386  AJMP ENDBS  ;DISABLE INTERRUPTS
387
0228 C2AF
388
389
022A 22
390  RET
391
392
393
394
395
396
397
398
399
400  $EJECT
**TEST GENERATION:** This subroutine generates **59** characters to be printed (20Hex to 5AHex) and saves them in internal RAM (locations 30 to 6AHex).

**TEST: HOW DO YOU SET STARTING ADDRESS**

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>LINE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>366</td>
<td>7930</td>
<td>TEST: MDV R1,$30H</td>
<td>SET STARTING ADDRESS</td>
</tr>
<tr>
<td>022B</td>
<td>7D20</td>
<td>MOV R5,$20H</td>
<td>STARTING CHARACTER=20HEX( )</td>
</tr>
<tr>
<td>022F</td>
<td>7E5A</td>
<td>MOV R6,$5AH</td>
<td>ENDING CHARACTER=5AHEX(Z)</td>
</tr>
<tr>
<td>0231</td>
<td>ED0</td>
<td>NXTST: MOV A1,R5</td>
<td>SAVE CHARACTER AT [R1]</td>
</tr>
<tr>
<td>0232</td>
<td>F70</td>
<td>MOV @R1,A</td>
<td>SAVE CHARACTER AT [R1]</td>
</tr>
<tr>
<td>0233</td>
<td>6E0</td>
<td>XRL A1,R6</td>
<td>SAVE CHARACTER AT [R1]</td>
</tr>
<tr>
<td>0234</td>
<td>6004</td>
<td>JZ TEND</td>
<td>JUMP IF &quot;Z&quot;</td>
</tr>
<tr>
<td>0236</td>
<td>090</td>
<td>INC R1</td>
<td>INCREMENT CHARACTER</td>
</tr>
<tr>
<td>0237</td>
<td>0D0</td>
<td>INC R5</td>
<td>GO TO NEXT CHARACTER</td>
</tr>
<tr>
<td>0238</td>
<td>4131</td>
<td>AJMP NXTST</td>
<td>SAVE NEXT CHARACTER</td>
</tr>
<tr>
<td>023A</td>
<td>796B</td>
<td>TEND: MOV R1,$6BH</td>
<td>PLACE TERMINATING ADDRESS INTO R1</td>
</tr>
<tr>
<td>023C</td>
<td>22</td>
<td>RET</td>
<td></td>
</tr>
<tr>
<td>388</td>
<td>;</td>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>

**ASSEMBLY COMPLETE, NO ERRORS FOUND**
BYTE TRANSFER ROUTINE: 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 CR13) to the printer.

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

;

TRAM:       MOV  A,R1 ;SAVE TERMINATING ADDRESS
            MOV  R7,A ;IN R7
            INC  R3
            MOV  TMDH,#16H ;SET TIMER AS 8-BIT COUNTER
            MOV  TLO,#0FFH ;SET LOW BYTE OF COUNTER TO FF
            MOV  THO,#0FFH ;SET RELOAD VALUE TO FF
            CLR  T0 ;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
            REPT: MOV  P1,#R1 ;OUTPUT RAM CONTENTS TO PRINTER
            INX  R9 ;INCREMENT ADDRESS
            MOVX  @RO,A ;GENERATE "DATA STROBE" (WR)
            INC  R1
            MOV  P1,#R1 ;OUTPUT NEXT RAM CONTENTS TO PRINTER
            MOV  A,R1
            JNB  TFO,’$ ;WAIT FOR "ACKNOWLEDGE" FROM PRINTER
            INTR:  CLR  TFO ;CLEAR OVERFLOW FLAG
                    XRL  A,R7 ;COMPARE CURRENT ADDR. TO TERMINAT ADDR
                    JNZ  NEXTB ;JUMP IF NOT EQUAL
                    MOV  R1,#30H ;RESET STARTING ADDRESS
                    DEC  R3 ;DECREMENT REPETITION $
                    CJNE  R3,#00H,REPT ;IF NOT EQUAL, 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
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
OR - Output Ready
SO - Shift Out
MR - Master Reset
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 O0-O3 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 O0-O3 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>
SHIFT OUT

SHIFT IN

INPUT READY

DATA THROUGHPUT AND INPUT READY TIMING

T12

T17

SHIFT IN

SHIFT OUT

OUTPUT READY

DATA THROUGHPUT AND OUTPUT READY TIMING

T12

T18
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.
A = 1.0 microsec. min.
B = 0.5 microsec. min.
C = 4 + 1.0 microsec.
D = Greater than zero
DATA TRANSFER TIMING WITHOUT BUSY

A = 1.0 microsec.min.  D = 4 ≥ 1.0 microsec.
B = 0.5 microsec.min.  E = Greater than zero
    = 1.0 microsec.max.  F = 0.0 microsec.min.
C = 7 ≥ 1.2 microsec.  = 0.075 microsec.max.
SELECT

DATA STROBE

BUSY

ACKNLG

DATA

Busy Condition

| Busy Duration | 1. Deselect code
| | Until printer is selected
| 2. CR W/O Auto Line | 7.0+1.0 microsec.
| 3. CR W/ Auto Line | Print + paper motion cycle
| 4. All Other Terminations | Print + paper motion cycle

DATA TRANSFER TIMING WITH BUSY
LIST OF REFERENCES


