Using the Intel 8085 Serial I/O Lines

John Wharton
Microcomputer Applications
Using the Intel 8085 Serial I/O Lines

Contents

INTRODUCTION ................................................. 1
   MCS Family Members ................................... 1
   Additional 8085 Instructions .......................... 1

CRT INTERFACE ............................................. 3
   Hardware Interface .................................... 3
   Software Package ...................................... 3
      OUTPUT ROUTINE .................................. 5
      INPUT ROUTINE .................................. 6
      TIMING ANALYSIS .................................. 6
      BAUD RATE IDENTIFICATION ROUTINE .............. 7

CASSETTE RECORDER INTERFACE .............................. 8
   Hardware Design ...................................... 8
   Software .............................................. 9
      OUTPUT ROUTINE .................................. 10
      INPUT ROUTINE .................................. 10

ADDITIONAL COMMENTS ...................................... 12

APPENDIX .................................................. 13
INTRODUCTION
This application note is intended to acquaint the reader with the Intel® MCS-85 family, and to explain how to use one of its key features, a direct serial data link between the CPU and the outside world. Two design examples will be provided: a versatile method for direct communications between the CPU and a CRT or other peripheral at any rate from 110 to 9600 baud, and a magnetic tape interface system which allows programs and data to be stored or loaded using a cheap audio cassette recorder. Both examples use software routines to replace extensive external hardware and to provide additional flexibility.

MCS Family Members
The MCS-85 family consists of the new 8085 N-channel, 8-bit microprocessor (Figure 1) and a variety of parts which provide memory, input/output, timing, and peripheral control capability.

![Figure 1. 8085 Pinout Diagram](image)

In many respects the 8085 can be thought of as simply a hardware refinement of the extremely popular Intel® 8080 processor, which was introduced in 1973. It is 100% software-compatible with its predecessor. In addition to being 50% faster, many of the external timing and control functions needed by the 8080A have been integrated into the 8085 (such as the 8224 Clock Generator and the 8228 System Controller), thus reducing the system part count. Additional features include four additional maskable and non-maskable interrupt pins, increased drive capability, and two new input and output lines. These pins, called SID and SOD, provide a serial I/O data link with the CPU. The 8085 uses a standard 40-pin DIP package.

All members of the MCS-85 family require a single 5-volt power supply, greatly reducing system overhead. Several components combine a number of system functions (e.g., ROM and I/O), allowing a complete, useful microcomputer system to be assembled with as few as three integrated circuits, as shown in Figure 2. For example, the 8155 and 8156 RAM/IO/TIMER chips each contain 256 8-bit bytes of program or data storage, three programmable I/O ports, and a 14-bit timer/counter.

Since the internal architecture and instruction set of the 8085 is an extension of that of the 8080, all software written for the 8080 – including compilers, assemblers, and individual applications programs – will run without modification on the new processor. In fact, the complete upward compatibility of the MCS-85 system means that applications designed around the 8080 can be converted to using the 8085 at minimal cost, requiring little hardware redesign or program modification. An engineer already familiar with the 8080 will not need to learn a new architecture or mnemonics.

Companies now using Intel's extensive line of design, development, and debugging tools can augment their systems with a series of 8085 support products (such as the SDK-85, ICE-85, and SBC boards) which are compatible with their present mainframe and peripherals.

Additional 8085 Instructions
The additional hardware features of the 8085 (handling multiple-level maskable interrupts and serial I/O) are supported with two new instructions. RIM (machine code 20H) is used to read the current status of the three interrupt masks into the accumulator. Additional instructions are set to show what interrupts (if any) are pending, and the logical state of the SID input pin (pin 5). The complement of RIM is SIM (machine code 30H), which has a dual function depending on the current accumulator contents. If bits 3 or 4 of the accumulator are a logical one, SIM can be used to change the three
Interrupt masks; if bit 6=1, SIM can set the SOD output (pin 4). The two functions of the SIM instruction operate independently. (If, at this point, the acronyms RIM, SIM, SID, and SOD are starting to blur in your mind, try to remember their roots instead: Read Interrupt Mask, Set Interrupt Mask, Serial Input Data, and Serial Output Data. Don't worry; of the other four ordered permutations of R&S, I&O, and M&D, only ROM is used elsewhere in this note, and then only in its traditional sense.)

![Diagram of 8085 Minimum System](image)

Figure 2. 8085 Minimum System

A detailed explanation of the accumulator contents after and before RIM and SIM is given in Figure 3. The I/O pins both function with respect to positive logic: a "1" in the accumulator corresponds to a high voltage level, "0" to a level near ground. The SID and SOD lines are electrically compatible with normal TTL logic levels. (For full electrical specifications, the reader should consult the MCS-85 User's Manual.) If A6 is "0" prior to executing SIM, SOD will remain unchanged, regardless of the state of A5. After a Reset, SOD will be low. It should be noted that RIM does not affect the Sign Flag; in order to make a conditional jump based on the Serial Input Data state, a three instruction sequence should be used, such as shown in Examples 1 and 2. When serial data is to be assembled into a parallel word, a sequence such as in Example 3 can be used, which shifts the contents of register pair HL one bit to the left and appends the input data bit.

![Diagram of RIM and SIM Instructions](image)

Figure 3. Effect of RIM and SIM Instructions

This note does not concern itself with the interrupt mask manipulation also made possible with RIM and SIM; for a full understanding of the interrupt capabilities of the 8085, see the User's Manual. To use SIM for altering the SOD state without fear of interfering with the interrupt mask status, one need only make sure that bits 3 and 4 of the accumulator are set to zero first.
zero “Start” bit, 8 data bits (least significant bit first), and one or more stop bits (logic 1). An 11-bit sequence with two stop bits is used for 110 baud TTY’s. The logic one level continues until the start bit of the next byte to ensure that each 10-bit sequence is initiated with a one-to-zero transition. The 8 bits transferred might be raw binary data or alphanumeric characters using the standard ASCII code. In this case, the most significant bit — the last data bit transmitted — will depend on the parity convention being used. This sequence is illustrated for the ASCII “space” character in Figure 4.

![Figure 4. ASCII Space Character](image)

On the following pages, examples are given showing two possible uses of the SID and SOD lines. The main purpose of these examples is not to give the reader a design after which he could model his own system — though, of course, this might be the case — but rather to illustrate the hardware and software interfaces and techniques necessary to implement a typical working subsystem.

**CRT INTERFACE**

Most microprocessor systems require some sort of serial communications. This may be selected for reasons of economy (to reduce the number of interconnections required in a distributed system), or it may be necessary in order to communicate with such common peripherals as CRT’s or typewriters.

These peripherals all use a standard convention for transmitting serial ASCII code. Each data byte is transmitted as a series of 10 or 11 bits. The uniform time per bit corresponds to the data transmission rate. For example, if the transmission rate is to be 2400 baud (2400 bits per second), each bit time must be 1/2400 bps = 416.7 μsec/bit. The standard 10-bit sequence consists of a logically
Upon power-up or reset, or when the console device baud rate is changed, the baud rate identification subroutine (BRID) is called. This routine waits until an ASCII space character (20H) is received from the console. (Any other character will result in a case of mistaken identification.) When a space character is received, two time parameters are computed which correspond to the bit time and one-half the bit time of the baud rate being used. These are stored as variables BITTIME and HALFBIT. To output a character to the console, the character code is placed in register C, and the subroutine COUT is called. This routine uses BITTIME as a parameter for the software delay loop which determines the baud rate. To accept a character from the keyboard, CIN is called. CIN returns after the next key is typed, with the corresponding character code in register C. CIN uses both parameters BITTIME and HALFBIT.

Since COUT and CIN use time parameters computed by BRID, they will function at a rate the same as that of the initial space character input. Because of the nature of the software, the rate does not depend on the CPU clock frequency. This results in additional flexibility in the following respects:

1. The software does not need to be modified if the 8085 crystal frequency is changed or Wait states are added.
2. Since the time base is no longer critical, the quartz crystal could be replaced by a less expensive RC network, provided the frequency does not drift by more than a few percent during a session. Additional drift can be accommodated by periodically recalling the BRID routine.
3. Communication is possible at non-standard baud rates which relaxes the constraints on system peripherals.

It should be noted, though, that slowing down the CPU clock will decrease its throughput proportionately. In addition, it will degrade the maximum resolution of the delay loops, with the result that the highest baud rates may no longer be achievable.

A more detailed analysis of the CRT interface routines will be presented in the order of increasing complexity: COUT, CIN, and BRID. Since SID and
SOD are ideal for many applications which involve critical I/O timing, the timing techniques used here may be of interest to software designers. Accordingly, the mathematical derivation of the timing parameters is included in this analysis, as well as a justification for the BRID algorithm. The algebra involved might be a bit too tedious for designers unconcerned with generating software delays. If so, they (and other bored readers) have the freedom of choice to skip over the sections they find objectionable.

**OUTPUT ROUTINE**

It would seem natural to write data in the standard format in three stages: output a zero start bit, then the 8 data bits (using a loop sequence), then the stop bits. Each stage would incorporate its own appropriate delay and output sections, leading to unnecessary duplication. Instead, the code below executes the same main loop 11 times. Its bit manipulation routine inherently results in the correct data sequence being formed. It accomplishes this by using the carry and C register as a 9-bit pseudo-circular shift register. Initially CY=0. The algorithm outputs CY, waits one bit time, sets CY=1, and then rotates the pseudo-register right one bit. This repeats for 11 cycles. On the tenth and all subsequent loops, the output bit will be a logical one, since that bit had been set nine loops earlier while in the CY (see Figure 6).

When COUT is called the registers to be used must be preserved and interrupts disabled so the timing loop will not be disrupted. Clear the CY in preparation for outputting the start bit, and set the loop counter for 11 bits (if 110 baud will never be used, the counter could be set to 10):

```plaintext
COUT: PUSH B
      PUSH H
      DI
      SRA A
      MVI B, 11
```

Output of the contents of the CY:

```plaintext
CO1: MVI A, 80H  (7)
      PAR  (4)
      SIM  (4)
```

The numbers in brackets indicate how many machine cycles are required for each instruction. They will be referred to in the timing analysis section.

Get stuck in a loop for the appropriate time (don’t worry for now how “BITTIME” is determined):

```plaintext
CO2: LLOAD BITTIME (16)
      DCR L
      JNZ CO2  (6)
      DCR H
      JNZ CO2  (6)
```

Rotate the contents of register C right into the CY, while moving a one into the left end. Continue until all bits have been transmitted:

```plaintext
STC
MOV A, C  (4)
SRA
MOV C, A  (4)
DCR B  (4)
JNZ CO1  (16)
```

Restore processor status and return:
INPUT ROUTINE

The console input routine uses the opposite procedure; instead of moving a bit from register C to the CY, then to A7, then to SOD, CIN loads a bit from SID into A7, then moves it to CY, then into register C.

First, set up the CPU as before:

\[
\begin{align*}
\text{CIN:} & \quad \text{PUSH} \quad H \\
\text{DI} & \\
\text{MOV} & \quad E, \quad 9
\end{align*}
\]

When a start bit transition arrives, the first sampling should not be taken until the middle of the first data bit, one and one-half bit times after the transition. Await the start bit transition, then set up the delay parameter for one-half bit time:

\[
\begin{align*}
\text{C11:} & \quad \text{RTH} \quad (4) \\
\text{OPH} & \quad H \quad (4) \\
\text{JM} & \quad \text{C11} \quad (7) \\
\text{LHLD} & \quad \text{HALFBIT} \quad (15)
\end{align*}
\]

Loop for one-half bit time before starting to sample data:

\[
\begin{align*}
\text{C12:} & \quad \text{DCR} \quad L \quad (6) \\
\text{JNZ} & \quad \text{C12} \quad (6) \\
\text{DCR} & \quad H \quad (6) \\
\text{JNZ} & \quad \text{C12} \quad (6)
\end{align*}
\]

Wait until the middle of the next bit before sampling SID, then move the data bit into CY:

\[
\begin{align*}
\text{C13:} & \quad \text{LHLD} \quad \text{BITTIME} \quad (15) \\
\text{C14:} & \quad \text{DCR} \quad L \quad (6) \\
\text{JNZ} & \quad \text{C14} \quad (6) \\
\text{DCR} & \quad H \quad (6) \\
\text{JNZ} & \quad \text{C14} \quad (6) \\
\text{RTH} & \quad (4) \\
\text{PFA} & \quad (4)
\end{align*}
\]

Decrement the bit counter. If this is the ninth cycle, the 8 data bits are in register C, so quit (the first stop bit will already have been received, and be in CY):

\[
\begin{align*}
\text{DOP} & \quad B \quad (4) \\
\text{JZ} & \quad \text{C15} \quad (7)
\end{align*}
\]

Otherwise, continue. Rotate the data bit right into register C, and repeat the cycle:

\[
\begin{align*}
\text{MOV} & \quad B, \quad C \quad (4) \\
\text{RAR} & \quad (4) \\
\text{MOV} & \quad C, \quad A \quad (4) \\
\text{NOP} & \quad (4) \\
\text{JMP} & \quad \text{C13} \quad (15)
\end{align*}
\]

(A NOP is needed to make the COUT and CIN loops exactly equal in number of machine cycles, so that each can use the same delay parameter.) Restore status and return.

\[
\begin{align*}
\text{C15:} & \quad \text{POP} \quad H \\
\text{EI} & \\
\text{RET}
\end{align*}
\]

TIMING ANALYSIS

COUT and CIN now need to be provided with parameters for BITTIME and HALFBIT. It can be seen from the above code that each routine uses \(61 + D\) machine cycles per input or output bit, where \(D\) is the number of cycles spent in either four line delay segment. If \((H)\) and \((L)\) are the contents of the \(H\) and \(L\) registers going into this section of code, then:

\[
D = 22 + ((L) - 1) \times 14 + ((H) - 1) \times [(255 \times 14) + 25]
\]

If \((H)' = (H) - 1, (L)' = (L) - 1, and (HL)' = 256 \times (H)' + (L)'\), then

\[
D = 22 + 14 \times (L)' + 3595 \times (H)'
\]

This can be approximated by:

\[
D = 22 + 14 \times (HL)'
\]

This approximation is exact for \((H)' = 0; otherwise, it is accurate to within 0.3%. Thus each loop of COUT or CIN uses a total of:

\[
C = 61 + D = 83 + 14 \times (HL)'
\]

Each machine cycle uses two crystal cycles in the 8085, so the resulting data rate is:

\[
B = \frac{\text{cycle frequency}}{C} = \frac{(\text{crystal frequency}) \div 2}{83 + 14 \times (HL)'}
\]
For a typical calculation, see Example 4.

**EXAMPLE 4**

To produce 2400 baud with the standard 6.144 MHz crystal:

\[
2400 = \frac{(6.144 \times 10^6) \pm 2}{83 + 14 (HL)}
\]

\[
14 (HL)' = \left( \frac{6.144 \times 10^6 \pm 2}{2400} \right) - 83
\]

\[
(HL)' = \left[ \frac{6.144 \times 10^6 \pm 2}{2400} \right] - 83
\]

\[
\frac{14}{HL} = 8610 = 0056H
\]

\[
(HL)' = 8610 = 0056H
\]

\[
(HL) = 0157H = \text{BITTIME}
\]

To determine the true data rate this parameter will produce, substitute into equation (6):

\[
\text{Data Rate} = \frac{6.144 \times 10^6 \pm 2}{83 + 14(86)}
\]

\[
= 2387 \text{ baud, which is 0.54% slow.}
\]

For 9600 baud, the same calculations will yield \((HL)\)' = 17, which is actually 0.3% slow; a snailish 19200 baud or 38400 baud could each be generated to within 5% if \((HL)\)' = 6 or 0! Table 1 presents the parameters for several standard baud rates.

Notice that the resolution of the delay algorithm—the difference between bit times resulting from parameters which differ by one—is 14 machine cycles. As a result, the true bit delay produced can always manage to be within ±2.3 µsec of the delay desired. This guarantees that at rates up to 9600 baud, where each bit time is at least 104 µsec wide, some value of BITTIME can be found which will be accurate to within 2.2%.

**BAUD RATE IDENTIFICATION ROUTINE**

The function of BRID is to compute the appropriate parameters BITTIME and HALFBIT. It accomplishes this by observing the data pattern received when the space bar is pressed on the console device. Since a space character has the ASCII code 20H = 00100000B, the pattern represented back in Figure 4 is transmitted. Notice that the initial zero level is 6 bits wide. Suppose it could be determined that this corresponds to \(M\) machine cycles. Then one bit would correspond to \((M=6)\) machine cycles. The reason for dividing down a space several bits long is so that any distortion caused by the signal rise and fall times, or any lack of precision in detecting the two transitions, will be reduced by a factor of six. Since the bit period of COUT and CIN is 83 + 14 \((HL)'\), BRID must generate a value \((HL)\) such that:

\[
M \div 6 = 83 + 14 (HL)'
\]

\[
(HL)' = \frac{(M \div 6) - 83}{14}
\]

\[
(HL)' = \frac{M}{84} - 6 \text{ (approximately)}
\]

This value can be determined by setting register pair HL to −6, then incrementing it once every 84 machine cycles during the period that the in-
ing signal is zero. BITTIME is then obtained by individually incrementing registers H and L. To obtain HALFBIT, divide the value of (HL) determined above by two before incrementing each register.

In order to implement this algorithm, set HL to -6, verify that the incoming signal is a logic one, then wait for the start bit bit transition.

```
BR10: MVI A, 00H
      SIM
      LMI H, -6H
BR11: RIM
      ORA R
      JP BR11
BR12: RIM
      ORA R
      JM BR12
```

Increment register pair HL, then delay so that each cycle will require 84 machine cycles:

```
BR13: INX H (6)
      MVI E, 04H (7)
BR14: DCR E (53)
      JNZ BR14 (7)
```

Check if SID is still low. If so, repeat:

```
RIM (4)
ORA R (4)
JP BR13 (10)
```

Otherwise continue. Store HL temporarily for the HALFBIT calculation. Obtain and store BITTIME:

```
PUSH H
INR H
INR L
SHLD BITTIME
```

Restore HL, calculate HALFBIT, and return:

```
POP H
ORA R
MOV A, H
RAR
MOV A, H
MOV A, L
RAR
MOV L, A
INR H
INR L
SHLD HALFBIT
RET
```

The assembled listings for these subroutines, along with a simple test program, is presented in the Appendix.

**Cassette Recorder Interface**

There are many situations where data has to be transmitted through a non-ideal medium. To give three typical examples, a system with electrically isolated elements might require that signals be AC coupled, communications through an audio network (such as telephone or radio) are greatly bandwidth limited, and some applications (such as a distributed network in an industrial environment) must tolerate random electrical noise. Attempting to record data on a cheap cassette recorder (the one used for this note cost $17.00) will reveal all of these shortcomings, plus one: The tape speed fluctuates significantly and varies as the batteries run down, hence the data rate is inconsistent.

The recording scheme used here makes very few demands on the transmission medium. It makes no attempt to transmit DC voltage levels. Instead, data is transmitted by a series of variable length tone bursts. The dominant frequency of the tone used can be selected to be within the passband of the particular medium. Data is transmitted with each bit composed of a tone burst followed by a pause. The first third of a bit period is always a tone burst, the middle third is either a tone burst continuous with the first or a pause corresponding to, respectively, a one or zero, and the final third is always a pause, as shown in Figure 7. Thus, data is distinguished by the burst/pause ratio.

**Hardware Design**

These tone bursts are obtained from the 8085 SOD line, using analog signal conditioning to eliminate the DC component of the waveform. (This low frequency component is due to the single-ended nature of the SOD line: its deviations from ground are all positive, which unbalances the capacitive input stage of the recorder.) A suggested interface circuit is shown in Figure 8, using one LM324 quad op amp and a few standard value discrete components which should be available in even a digital design laboratory. On playback, analog circuitry is again used to detect the presence of a tone burst. In Figure 8, A2 buffers the incoming signal, and A3 inverts it. The peaks of these two signals are transmitted through D1 or D2 and are filtered by an RC network. Comparator A4 then squares up the output and produces the logic signal read
by the SID pin. Since the op amps are powered by the single 5-volt supply, a 2.0-volt reference level is obtained from a resistive voltage divider. The waveforms present at several points in the circuit are shown in Figure 9.

**Software**

The algorithm for reading a data bit off the tape is simple and straightforward: If the tone burst is longer than the pause, the bit is a one. Otherwise, it is a zero. Since only the time ratio is considered, any variation in tape speed will not affect the data determination.

---

**VOLUME CONTROL**

A question that arises with any audio cassette interface is how to set the volume control. (Recording level is usually determined internally.) When the playback level is correct, the logic signal output from A4 will have either a one-third or two-thirds duty cycle. This can be readily observed with an oscilloscope. In the field, an old-fashioned mechanical-type voltmeter could be connected to the A4 output, and the volume adjusted until the meter needle hovered somewhere between 1/3 and 2/3 the high level output voltage. With random data, the reading would be about 2 volts. There will be a fairly wide range of acceptable volume settings. (Since the quivering meter needle is being used here for inertial signal averaging, a digital voltmeter would not be very helpful in this application.)

---

**Figure 7. Tape Interface Data Recording Scheme**

**Figure 8. One Chip Magnetic Tape Interface Schematic**
After the CRT software analysis, the tape routines are almost trivial. TAPEO is a subroutine for outputting the contents of register C to a cassette recorder. TAPEIN reads 8 bits into register C.

OUTPUT ROUTINE

TAPEO calls a subroutine named BURST three times for each bit. If A6 (the SOD enable bit) is set when BURST is called, a square-wave tone burst will be transmitted. If A6 is not set, BURST simply delays for exactly the same amount of time before returning. The three calls are used to, respectively, output the initial burst, output the data burst/space, and create the space at the end of each bit. Nine bits will be output: the eight data bits (LSB first) followed by a zero bit. The start of the initial burst of the trailing zero is needed to mark the end of the final space of the preceding data bit.

Start each bit by outputting a tone burst:

**TAPEO:**

```
MOV A, 8
```

**T01:**

```
MOV B, 00CH
CALL BURST
```

Rotate register C through CY:

```
MOV A, C
ROR
MOV C, A
```

Move CY to the SOD enable bit position, A6. Simultaneously set A7 to one, and clear all other bits. Output a tone burst or space, depending on the previous contents of CY:

```
MVI A, 01H
ROR
ROR
CALL BURST
```

Clear the accumulator, and output a space:

```
XRA A
CALL BURST
```

Keep cycling until the full 9-bit sequence is finished:

```
DCR B
JNZ T01
RET
```

The BURST subroutine executes the SIM instruction CYCNO times, at intervals of 29 + 14 (HALFCYC) machine cycles. In between each SIM, bit A7 is complemented. CYCNO should be an even number. If A6 is set upon calling BURST a square-wave will be created. Otherwise, the same code sequence is followed but SOD does not change – thus a space results.

**BURST:**

```
MVI D, CYCNO
```

**BU1:**

```
SIM
```

**BU2:**

```
MVI E, HALFCYC
```

```
DCR E
JNZ BU2
```

```
80H
```

```
DCR D
JNZ BU1
```

```
RET
```

INPUT ROUTINE

TAPEIN uses a subroutine called BITIN to move the data at the SID pin into the CY. The maximum rate at which SID is read is limited by a delay loop in BITIN.

Initialize the bit counter and the register D, which will keep track of the tone burst time. If a tone
burst is being received when TAPEIN is called, wait until the burst is over:

```
TAPEIN: MVI B, 8
         MVI D, 00H
T11:   CALL BITIN
         JC T11
         CALL BITIN
         JC T11
```

(Throughout this subroutine, a level transition is recognized only after it has been read once initially and then verified on the next reading. This provides some degree of software noise immunity.) Now wait the start of the next burst:

```
T12:   CALL BITIN
         JNC T12
         CALL BITIN
         JNC T12
```

The next burst has now arrived. Keep reading the SID pin, decrementing register D (thus making it more negative), each cycle until the pause is detected:

```
T13:   DCR D
         CALL BITIN
         JC T13
         CALL BITIN
         JC T13
```

Now continue reading the SID pin, incrementing the D register (back towards zero), each cycle until the next burst is received:

```
T14:   INR D
         CALL BITIN
         JNC T14
         CALL BITIN
         JNC T14
```

Now, if the burst lasted longer than the space, D was not incremented all the way back to zero; it is still negative. If the space was longer, D was incremented up through zero; it is now positive. In other words, the sign bit of D will now correspond to the data bit that would lead to each of these results. Move the sign bit into the CY, then rotate it into register C:

```
MOV A,D
RAL
MOV A,C
RAR
MOV C,A
MVI D, 00H
```

Continue until the last bit has been received:

```
OCR B
JNZ T13
RET
```

(Notice that the first half of this subroutine is incorporated in the second half. In fact, the assembled listing included in the Appendix makes use of this fact to eliminate 24 bytes of duplicated code.)

BITIN waits a short time in order to regulate the sampling rate, then reads SID and moves the data bit into the CY:

```
BITIN: MVI E, CKRATE <7>
B11:   DCR E <4>
         JNZ B11 <7/10>
         RIN <4>
         RAL <4>
         RET <10>
```

The tone burst frequency and duration, and the TAPEIN sampling rate are determined by HALFCYC, CYCNO, and CKRATE. Tables 2 and 3 give typical values.

**Table 2**

<table>
<thead>
<tr>
<th>APPROXIMATE TONE FREQUENCY</th>
<th>CORRESPONDING HALFCYC VALUE</th>
<th>RESULTING DATA RATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 Hz</td>
<td>217</td>
<td>42 17 3.3   bps</td>
</tr>
<tr>
<td>1 kHz</td>
<td>108</td>
<td>82 33 3.0   bps</td>
</tr>
<tr>
<td>2 kHz</td>
<td>53</td>
<td>166 66 13 bps</td>
</tr>
<tr>
<td>5 kHz</td>
<td>20</td>
<td>416 166 33 bps</td>
</tr>
<tr>
<td>10 kHz</td>
<td>9</td>
<td>826 330 65 bps</td>
</tr>
</tbody>
</table>

**Table 3**

<table>
<thead>
<tr>
<th>CKRATE VALUE</th>
<th>SAMPLING RATE (INCLUDING CALL &amp; RET)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17.6 µsec</td>
</tr>
<tr>
<td>20</td>
<td>104 µsec</td>
</tr>
<tr>
<td>80</td>
<td>378 µsec</td>
</tr>
<tr>
<td>250</td>
<td>1.14 µsec</td>
</tr>
</tbody>
</table>
The Appendix also includes a simple block record routine utilizing TAPEO. Before calling BLKRCDE, HL must be set to the start of the desired block, and the recorder turned on manually. Successive bytes will be recorded until the end of that page, i.e., until L is incremented to zero. The playback routine requires presetting HL to the target address and turning on the recorder before PLAYBK is called. These routines incorporate a long tone burst before each data block to allow a recorder with Automatic Gain Control to stabilize before the data starts.

ADDITIONAL COMMENTS

The two design examples given so far were built up using an SDK-85 System Design Kit. Both hardware interfaces were wire-wrapped on the ample breadboarding area provided on the board. The connections between SID and SOD and the on-board TTY interface were broken, so as not to affect the 8085 I/O electrical characteristics.

The CRT interface was tested with a Beehive Mini-Bee II Terminal in the full duplex mode at each of its 14 possible transmission rates, from 110 to 9600 baud. It was also checked at 19200 baud using a Beehive B-100 terminal. In addition, the software was exercised using an SBC 80/20 system as a variable baud rate character generator and receiver.

An additional advantage to having software selectable communications rates is that it would be possible to communicate with several system peripherals, each at its own preferred rate, without having to duplicate hardware. For example, the addition of a single 7408 AND gate and an output port (such as on the 8155) would make it possible to use the same two RS-232 circuits to interface with up to seven I/O devices (see Figure 10). Three of the MC1488 drivers have Enable inputs which can be controlled by the output port. One AND gate can be used to buffer the SOD line and drive the MC1488 Data inputs. The rest of the 7408 can be configured as a four input AND gate. This would act as an inverted logic OR gate to reduce the four MC1489 receiver outputs to a single line, which could be read by the SID. This assumes that only one input device (CRT, PTR) at a time will be used (which is usually the case in a non-time shared, interactive application), and that the unused devices are transmitting a logic one level (which should also be the case).

![Figure 10. Interfacing 8085 to Multiple Peripherals](image)

The software needed to support additional peripherals would be simple and straightforward. A routine intended to dump a section of memory to a paper tape punch, for example, would first have to store BITTIME and HALFBIT somewhere (perhaps on stack), load the variables with new parameters corresponding to the paper tape punch rate, and then write a bit pattern to the output port which would disable the console driver and enable the punch (and perhaps a typewriter). After the dump was over, the original time parameters and driver status would be restored.

As explained before, the BRID routine computed rate parameters based on the fact that an ASCII “space” character resulted in a zero level 6 bits long. Conceivably, some obscure peripherals might produce a transient between successive zero bits. (This might be the case, for example, if the signal was produced by mechanical rather than electronic means.) If so, the BRID algorithm used here probably would not work reliably. Once the two time parameters were identified, though, COUT and CIN could still be used. An alternate algorithm for baud rate identification would require a table in ROM (note the fifth and final R/S-I/O-M/D permutation). This table would contain a list of delay parameters corresponding to the standard transmis-
sion rates, as computed for the selected crystal frequency. Initialization would require the operator to hit a specific key several times (usually the "U" key, which generates a pattern of alternating ones and zeros). The identification routine would attempt to "read" this pattern at each baud rate, in turn, until finding the rate at which the read was successful.

The cassette recorder used to develop the tape interface was a Lloyd's push-button model which cost $17 in 1972. Empirical testing has indicated that for this application, the quality of the cassette recorder is less critical than the quality of the tape itself. In other words, some 33¢ cassettes were not very reliable, even when used with more expensive recorders.

When using a cassette at the beginning of a side, allow the tape to run for about 10 seconds until the leader has passed before starting to write data. Otherwise, data will be lost to the leader.

Depending on the recorder quality, the tone burst frequency and duration can be optimized for higher data rates by modifying HALFCYC and CYCNO. If so, CKRATE should also be reduced, so that between about 10 and 80 data samplings are made during a single (one-third width) tone burst. At greatly increased frequencies, some of the components in the analog interface might also be modified.

The two simple routines for recording and playing back blocks of data were intended to illustrate one way of using TAPEIN and TAPEO, and therefore do not contain any provisions for error detection or correction. Depending on the nature of a particular application, these routines could be augmented with parity bit or checksum comparison, or an error correcting code technique.

Funny things happen when recording and playing back a page of RAM which includes the subroutine stack. Eventually, PLAYK will start writing over the data at the top of the stack, destroying the subroutine traceback sequence. The next RET instruction will then cause a jump to a place where you'd rather not be.

The printout reproduced in the Appendix includes the assembled listings for the CRT and magnetic tape interfaces discussed in this application note. The object code produced was programmed into a 8755 EPROM, which was installed in the expansion PROM socket of the SDK-85 board. Some very minor differences exist between this listing and the code segments presented earlier, which were written for maximum clarity.

APPENDIX
THE FOLLOWING PROGRAMS AND SUBROUTINES ARE DESCRIBED IN DETAIL
IN INTEL CORPORATION'S APPLICATION NOTE AP-29, "USING THE 8085
SERIAL I/O LINES." THE FIRST SECTION IS A GENERAL PURPOSE CRT
INTERFACE WITH AUTOMATIC BAUD RATE IDENTIFICATION. THE SECOND
SECTION IS A MAGNETIC TAPE INTERFACE FOR STORING DATA ON CASSETTE
TAPE. THE CODE PRESENTED HERE IS ORIGINATED AT LOCATION 8000H.
AND MIGHT BE PART OF AN EXPANSION FROM IN AN INTEL SDK-85
SYSTEM DESIGN KIT.

ORG 8000H : STARTING ADDRESS OF SDK-85 EXPANSION FROM
CRTST: CRT INTERFACE TEST. WHEN CALLED, AWAITS THE SPACE BAR BEING PRESSED ON
THE SYSTEM CONSOLE. THEN RESPONDS WITH A DATA RATE VERIFICATION
MESSAGE. THERE AFTER, CHARACTERS TYPED ON THE KEYBOARD ARE ECHOED
ON THE DISPLAY TUBE. WHEN A BREAK KEY IS TYPED, THE ROUTINE IS
RE-STARTED, ALLOWING A DIFFERENT BAUD RATE TO BE SELECTED ON THE CRT.

BRID: BAUD RATE IDENTIFICATION SUBROUTINE
EXPECTS A <CR> (ASCII 0D) TO BE RECEIVED FROM THE CONSOLE
THE LENGTH OF THE INITIAL ZERO LEVEL (6 BITS WIDE) IS MEASURED
IN ORDER TO DETERMINE THE DATA RATE FOR FUTURE COMMUNICATIONS.
VERIFY THAT THE "ONE" LEVEL HAS BEEN ESTABLISHED
AS THE CRT IS POWERING UP
MONITOR SID LINE STATUS
LOOP UNTIL START BIT IS RECEIVED
BIBS COUNTER USED IN DETERMINING ZERO DURATION
MACHINE CYCLE DELAY LOOP
INCREMENT COUNTER EVERY 84 CYCLES WHILE SID IS LOW

THE FOLLOWING PROGRAMS AND SUBROUTINES ARE DESCRIBED IN DETAIL
IN INTEL CORPORATION'S APPLICATION NOTE AP-29, "USING THE 8085
SERIAL I/O LINES." THE FIRST SECTION IS A GENERAL PURPOSE CRT
INTERFACE WITH AUTOMATIC BAUD RATE IDENTIFICATION. THE SECOND
SECTION IS A MAGNETIC TAPE INTERFACE FOR STORING DATA ON CASSETTE
TAPE. THE CODE PRESENTED HERE IS ORIGINATED AT LOCATION 8000H.
AND MIGHT BE PART OF AN EXPANSION FROM IN AN INTEL SDK-85
SYSTEM DESIGN KIT.

ORG 8000H : STARTING ADDRESS OF SDK-85 EXPANSION FROM
CRTST: CRT INTERFACE TEST. WHEN CALLED, AWAITS THE SPACE BAR BEING PRESSED ON
THE SYSTEM CONSOLE. THEN RESPONDS WITH A DATA RATE VERIFICATION
MESSAGE. THERE AFTER, CHARACTERS TYPED ON THE KEYBOARD ARE ECHOED
ON THE DISPLAY TUBE. WHEN A BREAK KEY IS TYPED, THE ROUTINE IS
RE-STARTED, ALLOWING A DIFFERENT BAUD RATE TO BE SELECTED ON THE CRT.

BRID: BAUD RATE IDENTIFICATION SUBROUTINE
EXPECTS A <CR> (ASCII 0D) TO BE RECEIVED FROM THE CONSOLE
THE LENGTH OF THE INITIAL ZERO LEVEL (6 BITS WIDE) IS MEASURED
IN ORDER TO DETERMINE THE DATA RATE FOR FUTURE COMMUNICATIONS.
VERIFY THAT THE "ONE" LEVEL HAS BEEN ESTABLISHED
AS THE CRT IS POWERING UP
MONITOR SID LINE STATUS
LOOP UNTIL START BIT IS RECEIVED
BIBS COUNTER USED IN DETERMINING ZERO DURATION
MACHINE CYCLE DELAY LOOP
INCREMENT COUNTER EVERY 84 CYCLES WHILE SID IS LOW

THE FOLLOWING PROGRAMS AND SUBROUTINES ARE DESCRIBED IN DETAIL
IN INTEL CORPORATION'S APPLICATION NOTE AP-29, "USING THE 8085
SERIAL I/O LINES." THE FIRST SECTION IS A GENERAL PURPOSE CRT
INTERFACE WITH AUTOMATIC BAUD RATE IDENTIFICATION. THE SECOND
SECTION IS A MAGNETIC TAPE INTERFACE FOR STORING DATA ON CASSETTE
TAPE. THE CODE PRESENTED HERE IS ORIGINATED AT LOCATION 8000H.
AND MIGHT BE PART OF AN EXPANSION FROM IN AN INTEL SDK-85
SYSTEM DESIGN KIT.

ORG 8000H : STARTING ADDRESS OF SDK-85 EXPANSION FROM
CRTST: CRT INTERFACE TEST. WHEN CALLED, AWAITS THE SPACE BAR BEING PRESSED ON
THE SYSTEM CONSOLE. THEN RESPONDS WITH A DATA RATE VERIFICATION
MESSAGE. THERE AFTER, CHARACTERS TYPED ON THE KEYBOARD ARE ECHOED
ON THE DISPLAY TUBE. WHEN A BREAK KEY IS TYPED, THE ROUTINE IS
RE-STARTED, ALLOWING A DIFFERENT BAUD RATE TO BE SELECTED ON THE CRT.

BRID: BAUD RATE IDENTIFICATION SUBROUTINE
EXPECTS A <CR> (ASCII 0D) TO BE RECEIVED FROM THE CONSOLE
THE LENGTH OF THE INITIAL ZERO LEVEL (6 BITS WIDE) IS MEASURED
IN ORDER TO DETERMINE THE DATA RATE FOR FUTURE COMMUNICATIONS.
VERIFY THAT THE "ONE" LEVEL HAS BEEN ESTABLISHED
AS THE CRT IS POWERING UP
MONITOR SID LINE STATUS
LOOP UNTIL START BIT IS RECEIVED
BIBS COUNTER USED IN DETERMINING ZERO DURATION
MACHINE CYCLE DELAY LOOP
INCREMENT COUNTER EVERY 84 CYCLES WHILE SID IS LOW

THE FOLLOWING PROGRAMS AND SUBROUTINES ARE DESCRIBED IN DETAIL
IN INTEL CORPORATION'S APPLICATION NOTE AP-29, "USING THE 8085
SERIAL I/O LINES." THE FIRST SECTION IS A GENERAL PURPOSE CRT
INTERFACE WITH AUTOMATIC BAUD RATE IDENTIFICATION. THE SECOND
SECTION IS A MAGNETIC TAPE INTERFACE FOR STORING DATA ON CASSETTE
TAPE. THE CODE PRESENTED HERE IS ORIGINATED AT LOCATION 8000H.
AND MIGHT BE PART OF AN EXPANSION FROM IN AN INTEL SDK-85
SYSTEM DESIGN KIT.

ORG 8000H : STARTING ADDRESS OF SDK-85 EXPANSION FROM
CRTST: CRT INTERFACE TEST. WHEN CALLED, AWAITS THE SPACE BAR BEING PRESSED ON
THE SYSTEM CONSOLE. THEN RESPONDS WITH A DATA RATE VERIFICATION
MESSAGE. THERE AFTER, CHARACTERS TYPED ON THE KEYBOARD ARE ECHOED
ON THE DISPLAY TUBE. WHEN A BREAK KEY IS TYPED, THE ROUTINE IS
RE-STARTED, ALLOWING A DIFFERENT BAUD RATE TO BE SELECTED ON THE CRT.

BRID: BAUD RATE IDENTIFICATION SUBROUTINE
EXPECTS A <CR> (ASCII 0D) TO BE RECEIVED FROM THE CONSOLE
THE LENGTH OF THE INITIAL ZERO LEVEL (6 BITS WIDE) IS MEASURED
IN ORDER TO DETERMINE THE DATA RATE FOR FUTURE COMMUNICATIONS.
VERIFY THAT THE "ONE" LEVEL HAS BEEN ESTABLISHED
AS THE CRT IS POWERING UP
MONITOR SID LINE STATUS
LOOP UNTIL START BIT IS RECEIVED
BIBS COUNTER USED IN DETERMINING ZERO DURATION
MACHINE CYCLE DELAY LOOP
INCREMENT COUNTER EVERY 84 CYCLES WHILE SID IS LOW

THE FOLLOWING PROGRAMS AND SUBROUTINES ARE DESCRIBED IN DETAIL
IN INTEL CORPORATION'S APPLICATION NOTE AP-29, "USING THE 8085
SERIAL I/O LINES." THE FIRST SECTION IS A GENERAL PURPOSE CRT
INTERFACE WITH AUTOMATIC BAUD RATE IDENTIFICATION. THE SECOND
SECTION IS A MAGNETIC TAPE INTERFACE FOR STORING DATA ON CASSETTE
TAPE. THE CODE PRESENTED HERE IS ORIGINATED AT LOCATION 8000H.
AND MIGHT BE PART OF AN EXPANSION FROM IN AN INTEL SDK-85
SYSTEM DESIGN KIT.

ORG 8000H : STARTING ADDRESS OF SDK-85 EXPANSION FROM
CRTST: CRT INTERFACE TEST. WHEN CALLED, AWAITS THE SPACE BAR BEING PRESSED ON
THE SYSTEM CONSOLE. THEN RESPONDS WITH A DATA RATE VERIFICATION
MESSAGE. THERE AFTER, CHARACTERS TYPED ON THE KEYBOARD ARE ECHOED
ON THE DISPLAY TUBE. WHEN A BREAK KEY IS TYPED, THE ROUTINE IS
RE-STARTED, ALLOWING A DIFFERENT BAUD RATE TO BE SELECTED ON THE CRT.

BRID: BAUD RATE IDENTIFICATION SUBROUTINE
EXPECTS A <CR> (ASCII 0D) TO BE RECEIVED FROM THE CONSOLE
THE LENGTH OF THE INITIAL ZERO LEVEL (6 BITS WIDE) IS MEASURED
IN ORDER TO DETERMINE THE DATA RATE FOR FUTURE COMMUNICATIONS.
VERIFY THAT THE "ONE" LEVEL HAS BEEN ESTABLISHED
AS THE CRT IS POWERING UP
MONITOR SID LINE STATUS
LOOP UNTIL START BIT IS RECEIVED
BIBS COUNTER USED IN DETERMINING ZERO DURATION
MACHINE CYCLE DELAY LOOP
INCREMENT COUNTER EVERY 84 CYCLES WHILE SID IS LOW

THE FOLLOWING PROGRAMS AND SUBROUTINES ARE DESCRIBED IN DETAIL
IN INTEL CORPORATION'S APPLICATION NOTE AP-29, "USING THE 8085
SERIAL I/O LINES." THE FIRST SECTION IS A GENERAL PURPOSE CRT
INTERFACE WITH AUTOMATIC BAUD RATE IDENTIFICATION. THE SECOND
SECTION IS A MAGNETIC TAPE INTERFACE FOR STORING DATA ON CASSETTE
TAPE. THE CODE PRESENTED HERE IS ORIGINATED AT LOCATION 8000H.
AND MIGHT BE PART OF AN EXPANSION FROM IN AN INTEL SDK-85
SYSTEM DESIGN KIT.

ORG 8000H : STARTING ADDRESS OF SDK-85 EXPANSION FROM
CRTST: CRT INTERFACE TEST. WHEN CALLED, AWAITS THE SPACE BAR BEING PRESSED ON
THE SYSTEM CONSOLE. THEN RESPONDS WITH A DATA RATE VERIFICATION
MESSAGE. THERE AFTER, CHARACTERS TYPED ON THE KEYBOARD ARE ECHOED
ON THE DISPLAY TUBE. WHEN A BREAK KEY IS TYPED, THE ROUTINE IS
RE-STARTED, ALLOWING A DIFFERENT BAUD RATE TO BE SELECTED ON THE CRT.

BRID: BAUD RATE IDENTIFICATION SUBROUTINE
EXPECTS A <CR> (ASCII 0D) TO BE RECEIVED FROM THE CONSOLE
THE LENGTH OF THE INITIAL ZERO LEVEL (6 BITS WIDE) IS MEASURED
IN ORDER TO DETERMINE THE DATA RATE FOR FUTURE COMMUNICATIONS.
VERIFY THAT THE "ONE" LEVEL HAS BEEN ESTABLISHED
AS THE CRT IS POWERING UP
MONITOR SID LINE STATUS
LOOP UNTIL START BIT IS RECEIVED
BIBS COUNTER USED IN DETERMINING ZERO DURATION
MACHINE CYCLE DELAY LOOP
INCREMENT COUNTER EVERY 84 CYCLES WHILE SID IS LOW
LOC OBJ SEG SOURCE STATEMENT
082F B7 54 ORA A
0830 F22708 55 JP BRI3
 56 ;CHR NOW CORRESPONDS TO INCOMING DATA RATE
0833 E5 57 PUSH H ;SAVE COUNT FOR HALF-BIT TIME COMPUTATION
0834 24 58 INR H ;BITTIME IS DETERMINED BY INCREMENTING
0835 2C 59 INR L \ H AND L INDIVIDUALLY
0836 220820 60 SHLD BITTIME
0839 E1 61 POP H ;RESTORE COUNT FOR HALF-BIT DETERMINATION
083A B7 62 ORA A ;CLEAR CARRY
083B 7C 63 MOV A.H ;ROTATE RIGHT EXTENDED (CHR)
083C 1F 64 RAR \ \ TO DIVIDE COUNTER BY 2
083D 67 65 MOV H.A
083E 7D 66 MOV A.L
0840 6F 68 MOV L.A
0841 24 69 INR H ;PUT H AND L IN PROPER FORMAT FOR DELAY
0842 2C 70 INR L \ SEGMENTS (INCREMENT EACH)
0843 220820 71 SHLD HALF-BIT ;SAVE AS HALF-BIT TIME DELAY PARAMETER
0846 C9 72 RET

74 ;SIGN ON WRITES A SIGN-ON MESSAGE TO THE CRT AT WHAT SHOULD BE THE CORRECT RATE.
75 ; IF THE MESSAGE IS UNINTELLIGIBLE . . . WELL, SO IT GOES.
0847 215588 76 SIGN: LVI H.STRING ;LOAD START OF SIGN-ON MESSAGE
084A 4E 77 S1: MOV C.M ;SET NEXT CHARACTER
084B 48 78 XRA A ;CLEAR ACCUMULATOR
084C B1 79 ORA C ;CHECK IF CHARACTER IS END OF STRING
084D C8 80 RZ ;RETURN IF SIGN-ON COMPLETE
084E D06998 81 CALL COUNT ELSE OUTPUT CHARACTER TO CRT
0851 23 82 INX H ;INDEX POINTER
0852 C34A08 83 JMP S1 ;ECHO NEXT CHARACTER

84
0855 0D 85 STRING: DB 0DH, 0AH ;COR+CLF>
0856 0A
0857 42413544 86 DB 'BAUD RATE CHECK'
085B 20524154
085F 45204348
0863 454348
0865 0D 87 DB 0DH, 0AH ;COR+CLF>
0867 0A
0868 00 88 DB 00H ;END-OF-STRING ESCAPE CODE

89
90 ;COUT CONSOLE OUTPUT SUBROUTINE
91 ; WRITES THE CONTENTS OF THE C REGISTER TO THE CRT DISPLAY SCREEN
0869 F3 92 COUT: DI
086A C5 93 PUSH D
086B E5 94 PUSH H
086C 0600 95 MVI B, 8.BITSO ;SET NUMBER OF BITS TO BE TRANSMITTED
086E AF 96 XRA A ;CLEAR CARRY
086F 3E80 97 CO1: MVI A, 80H ;SET WHAT WILL BECOME SOD ENABLE BIT
0871 1F 98 RAR ;MOVE CARRY INTO SOD DATA BIT OF ACC
0872 39 99 SIM ;OUTPUT DATA BIT TO SOD
0873 2AC320 100 LHLD BITTIME
0876 20 101 CO2: DCR L ;WAIT UNTIL APPROPRIATE TIME HAS PASSED
LOC OBJ  SE0  SOURCE STATEMENT
0077 C27608 102  JNZ  CO2
0078 25  103  DCR  H
0079 C27608 104  JNZ  CO2
007F 37  105  STC  SET WHAT WILL EVENTUALLY BECOME A STOP BIT
007F 79  106  MOV  A,C  ROTATE CHARACTER RIGHT ONE BIT
0080 1F  107  PRR  MOVING NEXT DATA BIT INTO CARRY
0081 4F  108  MOV  C,A
0082 05  109  DCR  B  CHECK IF CHARACTER (AND STOP BIT(S)) DONE
0083 C26F08 110  JNZ  CO1  IF NOT, OUTPUT CURRENT CARRY
0086 E1  111  POP  H  RESTORE STATUS AND RETURN
0087 C1  112  POP  B
0088 FB  113  EI
0089 C9  114  RET
115
116 ;CIN  CONSOL INPUT SUBROUTINE WAITS FOR A KEYSTROKE AND
117 ;  RETURNS WITH 8 BITS IN REG C.
008A F3  118  CIN:  DI
008B E5  119  PUSH  H
008C 0609  120  MVI  B.BITS: DATA BITS TO BE READ (LAST RETURNED IN CY).
008E 20  121  CII:  RIM  WAIT FOR SYNC BIT TRANSITION
008F B7  122  ORA  A
0090 FA8E08  123  JMP  C11
0093 2AC920  124  LHLD  HALFBIT
0095 2D  125  C12:  DCR  L  WAIT UNTIL MIDDLE OF START BIT
0097 C29608  126  JNZ  C12
0099 25  127  DCR  H
009A C29608  128  JNZ  C12
009C 2AC920  129  C13:  LHLD  BITTIME; WAIT OUT BIT TIME
00A0 2D  130  C14:  DCR  L
00A2 C2A100  131  JNZ  C14
00A5 25  132  DCR  H
00A6 C2A100  133  JNZ  C14
00A9 20  134  RIM  :CHECK SID LINE LEVEL
00AA 17  135  RAL  DATA BIT IN CY
00AB 05  136  DCR  B  DETERMINE IF THIS IS FIRST STOP BIT
00AC CB8508  137  JCZ  C15:  IF SO, JUMP OUT OF LOOP
00AF 79  138  MOV  A,C  ELSE ROTATE INTO PARTIAL CHARACTER IN C
00A0 1F  139  PRR  ACC HOLDS UPDATED CHARACTER
00B1 4F  140  MOV  C,A
00B2 00  141  JMP  C13
00B3 C29608  142  JMP  C13
00B6 E1  143  C15:  POP  H
00B7 FB  144  EI
00B8 C9  145  RET  :CHARACTER COMPLETE
146
147 ;**********************************************************************************************
148
149 ;  THE FOLLOWING CODE IS USED BY THE CASSETTE INTERFACE.
150 ;  SUBROUTINES TAPED AND TAPEIN ARE USED RESPECTIVELY
151 ;  TO OUTPUT OR RECEIVE AN EIGHT BIT BYTE OF DATA. REGISTER C
152 ;  HOLDS THE DATA IN EITHER CASE. REGISTERS A,B,C,A ARE ALL DESTROYED.
00C8 153 CYNCDE EQU 16  :AID THE NUMBER OF CYCLES PER TONE BURST
00C9 154 HALFCYC EQU 30  :DETERMINES TONE FREQUENCY
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJ</th>
<th>SEQ</th>
<th>Source Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0016</td>
<td>CSRE</td>
<td>EQU 22</td>
<td>SETS SAMPLE RATE</td>
</tr>
<tr>
<td>006A</td>
<td>LEADE</td>
<td>EQU 250</td>
<td>NUMBER OF SUCCESSIVE TONE BURSTS COMPRISING LEADER</td>
</tr>
<tr>
<td>006A</td>
<td>LRACH</td>
<td>EQU 250</td>
<td>USED IN PLANBK TO VERIFY PRESENCE OF LEADER</td>
</tr>
<tr>
<td>158</td>
<td>BLX/RCD</td>
<td></td>
<td>OUTPUTS A VERY LONG TONE BURST (LEADER) TIMES</td>
</tr>
<tr>
<td>160</td>
<td></td>
<td></td>
<td>THE NORMAL BURST DURATION TO ALLOW Recorder ELECTRONICS</td>
</tr>
<tr>
<td>161</td>
<td></td>
<td></td>
<td>AND AGC TO STABILIZE, THEN OUTPUTS THE REMAINDER OF THE</td>
</tr>
<tr>
<td>162</td>
<td></td>
<td></td>
<td>256 BYTE PAGE POINTED TO BY $2D, STARTING AT BYTE (L).</td>
</tr>
<tr>
<td>0088</td>
<td>BFEA</td>
<td>163 BLX/RCD;</td>
<td>MVI C: LEADER SET UP LEADER BURST LENGTH</td>
</tr>
<tr>
<td>0088</td>
<td>3EC8</td>
<td>164 MVI A: O08H:</td>
<td>SET ACCUMULATOR TO RESULT IN TONE BURST</td>
</tr>
<tr>
<td>0088</td>
<td>CDFF08</td>
<td>165 BR1:</td>
<td>CALL BURST OUTPUT TONE</td>
</tr>
<tr>
<td>0088</td>
<td>0D</td>
<td>166 DCR</td>
<td>C</td>
</tr>
<tr>
<td>0088</td>
<td>2D08</td>
<td>167 JNZ BR1:</td>
<td>SUSTAIN LEADER TONE</td>
</tr>
<tr>
<td>0088</td>
<td>AF</td>
<td>168 XRA</td>
<td>A: CLEAR ACCUMULATOR &amp; OUTPUT SPACE, SO THAT</td>
</tr>
<tr>
<td>0088</td>
<td>CDFF08</td>
<td>169 CALL</td>
<td>BURST ANY START OF FIRST DATA BYTE CAN BE DETECTED</td>
</tr>
<tr>
<td>0088</td>
<td>4E</td>
<td>170 BR2:</td>
<td>MOV C, M: GET DATA BYTE TO BE RECORDED</td>
</tr>
<tr>
<td>0088</td>
<td>CD0808</td>
<td>171 CALL</td>
<td>TAPE: OUTPUT REGISTER C TO RECORDER</td>
</tr>
<tr>
<td>0088</td>
<td>2C</td>
<td>172 INR</td>
<td>L: POINT TO NEXT BYTE</td>
</tr>
<tr>
<td>0088</td>
<td>C20808</td>
<td>173 JNZ BR2</td>
<td></td>
</tr>
<tr>
<td>0088</td>
<td>C9</td>
<td>174 RET</td>
<td>AFTER BLOCK IS COMPLETE</td>
</tr>
<tr>
<td>175</td>
<td></td>
<td></td>
<td>TAPEO: OUTPUTS THE BYTE IN REGISTER C TO THE RECORDER</td>
</tr>
<tr>
<td>176</td>
<td></td>
<td></td>
<td>REGISTERS A.B.C.D.EF ARE ALL USED.</td>
</tr>
<tr>
<td>0088</td>
<td>F3</td>
<td>177 TAPEO:</td>
<td>DI</td>
</tr>
<tr>
<td>0088</td>
<td>05</td>
<td>186 PUSH</td>
<td>D: D08E USED AS COUNTERS BY SUBROUTINE BURST</td>
</tr>
<tr>
<td>0088</td>
<td>09</td>
<td>181 MVI B: 9:</td>
<td>WILL RESULT IN 8 DATA BITS AND ONE STOP BIT</td>
</tr>
<tr>
<td>0088</td>
<td>AF</td>
<td>192 TO1:</td>
<td>XRA A: CLEAR ACCUMULATOR</td>
</tr>
<tr>
<td>0088</td>
<td>3EC8</td>
<td>193 MVI A: O08H:</td>
<td>SET ACCUMULATOR TO CAUSE A TONE BURST</td>
</tr>
<tr>
<td>0088</td>
<td>CDFF08</td>
<td>194 CALL</td>
<td>BURST</td>
</tr>
<tr>
<td>0088</td>
<td>7F</td>
<td>195 MOV</td>
<td>A.C: MOVE NEXT DATA BIT INTO THE CARRY</td>
</tr>
<tr>
<td>0088</td>
<td>1F</td>
<td>196 RAR</td>
<td></td>
</tr>
<tr>
<td>0088</td>
<td>4F</td>
<td>197 MOV</td>
<td>C, A: CARRY WILL BECOME S0D ENABLE IN BURST ROUTINE</td>
</tr>
<tr>
<td>0088</td>
<td>601</td>
<td>198 MVI A: 01H:</td>
<td>SET BIT TO BE REPEATEDLY COMPLEMENTED IN BURST</td>
</tr>
<tr>
<td>0088</td>
<td>1F</td>
<td>199 RAR</td>
<td></td>
</tr>
<tr>
<td>0088</td>
<td>1F</td>
<td>200 RAR</td>
<td></td>
</tr>
<tr>
<td>0088</td>
<td>CDFF08</td>
<td>201 CALL</td>
<td>BURST: OUTPUT EITHER A TONE OR A Paige</td>
</tr>
<tr>
<td>0088</td>
<td>AF</td>
<td>202 XRA</td>
<td>A: CLEAR ACCUMULATOR</td>
</tr>
<tr>
<td>0088</td>
<td>CDFF08</td>
<td>203 CALL</td>
<td>BURST: OUTPUT PAUSE</td>
</tr>
<tr>
<td>0088</td>
<td>05</td>
<td>204 DCR</td>
<td>B</td>
</tr>
<tr>
<td>0088</td>
<td>C20808</td>
<td>205 JNZ TO1:</td>
<td>REPEAT UNTIL BYTE FINISHED</td>
</tr>
<tr>
<td>0088</td>
<td>D1</td>
<td>206 POP</td>
<td>D: RESTORE STATUS AND RETURN</td>
</tr>
<tr>
<td>0088</td>
<td>FB</td>
<td>197 EI</td>
<td></td>
</tr>
<tr>
<td>0088</td>
<td>C9</td>
<td>198 RET</td>
<td></td>
</tr>
<tr>
<td>199</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0088</td>
<td>1610</td>
<td>200 BURST:</td>
<td>MVI D: CYCNO: SET NUMBER OF CYCLES</td>
</tr>
<tr>
<td>0088</td>
<td>30</td>
<td>201 BUL:</td>
<td>SIM: COMPLEMENT S0D LINE IF S0D ENABLE BIT SET</td>
</tr>
<tr>
<td>0088</td>
<td>1E1</td>
<td>202 MVI E: HALFCYC</td>
<td></td>
</tr>
<tr>
<td>0088</td>
<td>1D</td>
<td>203 BU2:</td>
<td>DCR E: REGULATE TONE FREQUENCY</td>
</tr>
<tr>
<td>0088</td>
<td>C2F08</td>
<td>204 JNZ BU2</td>
<td></td>
</tr>
<tr>
<td>0088</td>
<td>EE80</td>
<td>205 XRI 80H:</td>
<td>COMPLEMENT S0D DATA BIT IN ACCUMULATOR</td>
</tr>
<tr>
<td>0088</td>
<td>15</td>
<td>206 DCR</td>
<td>D</td>
</tr>
<tr>
<td>0088</td>
<td>C2F08</td>
<td>207 JNZ BU2:</td>
<td>CONTINUE UNTIL BURST (OR EQUIVALENT PAUSE) FINISHED</td>
</tr>
</tbody>
</table>
LOC  OBJ  SEQ  SOURCE STATEMENT

08FF C9  268  RET
   269
   270; PLAYBK: WAITS FOR THE LONG LEADER BURST TO ARRIVE, THEN CONTINUES
   271; READING BYTES FROM THE RECORDER AND STORING THEM
   272; IN MEMORY STARTING AT LOCATION $07H.
   273; CONTINUES UNTIL THE END OF THE CURRENT PAGE ($7F<0FFH) IS REACHED.
0900 0EFA  214  PLAYBK: MVI C.LORCH: CLORCH: SUCCESSIVE HIGHS MUST BE READ
0902 CD3D09  215  P81: CALL BITIN: \ TO VERIFY THAT THE LEADER IS PRESENT
0905 D20009  216  JNZ PLAYBK: \ AND ELECTRONICS HAS STABILIZED
0906 80  217  DCR C
0909 C30209  218  JNZ PB1
090C CD3D09  219  PB2: CALL TAPEIN: \ GET DATA BYTE FROM RECORDER
090F 71  220  MOV M.C: STORE IN MEMORY
0910 2C  221  INR L: INCREMENT POINTER
0911 C30209  222  JNZ PB2: \ REPEAT FOR REST OF CURRENT PAGE
0914 C9  223  RET
   224
   225; TAPEIN CASSETTE TAPE INPUT SUBROUTINE. READS ONE BYTE OF DATA
   226; FROM THE RECORDER INTERFACE AND RETURNS WITH THE BYTE IN REGISTER C.
0915 0E99  227  TAPEIN: MVI B,9: READ EIGHT DATA BITS
0917 1800  228  T11: MVI D,00H: CLEAR UP/DOWN COUNTER
0919 15  229  T12: DCR D: DECREMENT COUNTER EACH TIME ONE LEVEL IS READ
091A CD3D09  230  CALL BITIN
091D DA19609  231  JC T12: \ REPEAT IF STILL AT ONE LEVEL
0920 CD3D09  232  CALL BITIN
0923 DA19609  233  JC T12
0926 14  234  T12: INR D: INCREMENT COUNTER EACH TIME ZERO IS READ
0927 CD3D09  235  CALL BITIN
092A D22609  236  JNC T13: \ REPEAT EACH TIME ZERO IS READ
092D CD3D09  237  CALL BITIN
0930 D22609  238  JNC T13
0933 7A  239  MOV R,A
0934 17  240  RAL: MOVE COUNTER MOST SIGNIFICANT BIT INTO CARRY
0935 79  241  MOVR C
0936 1F  242  PRL: MOVE DATA BIT RECEIVED (CY) INTO BYTE REGISTER
0937 4F  243  MOV C,A
0938 05  244  DCR B
0939 CD1709  245  JNZ T11: \ REPEAT UNTIL FULL BYTE ASSEMBLED
093C C3  246  RET
   247
   249
093D 1E16  248  BITIN: MVI E,CKRATE
093F 1D  249  B11: DCR E
0940 C2FA09  250  JNZ B11: LIMIT INPUT SAMPLING RATE
0943 20  251  RIN: SAMPLE SID LINE
0944 17  252  RAL: MOVE DATA INTO CY EIT
0945 C9  253  RET
   254
   255  END

PUBLIC SYMBOLS
EXTERNAL SYMBOLS

USER SYMBOLS
BL1 A 093F BITIN A 093D BITSI A 0009 BITSO A 0008 BITTM A 20C8 BLXRCD A 0089 BR1 A 008D
BR2 A 0008 BRI1 A 061F BRI2 A 0627 BRI4 A 0629 BRID A 001A BU1 A 00F2 BU2 A 00F5
BURST A 00F0 C1L A 008E C12 A 0096 C13 A 009E C14 A 0081 CI5 A 0086 CIN A 008A
CKRATE A 0016 C01 A 006F C02 A 0076 COUT A 0069 CRT1 A 0003 CRTIST A 0000 CYCNO A 0010
ECHO A 000C HALF21 A 20CA HALF2Y A 001E LDRCHK A 00FA LEADER A 00FA PB1 A 0002 PB2 A 000C
PLAYK A 0080 SL A 084A SIGNON A 0047 STRNG A 0055 TAEIN A 0915 TAEPO A 0801 T11 A 0917
TI2 A 0919 T12 A 0926 TO1 A 0005

ASSEMBLY COMPLETE. NO ERROR(S)
<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>249#</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td>BITIN</td>
<td>215</td>
<td>250</td>
</tr>
<tr>
<td>BIT51</td>
<td>16#</td>
<td>120</td>
</tr>
<tr>
<td>BIT50</td>
<td>15#</td>
<td>95</td>
</tr>
<tr>
<td>BITIN</td>
<td>13#</td>
<td>60</td>
</tr>
<tr>
<td>BLKPCD</td>
<td>16#</td>
<td>129</td>
</tr>
<tr>
<td>BR1</td>
<td>165#</td>
<td>157</td>
</tr>
<tr>
<td>BR2</td>
<td>170#</td>
<td>173</td>
</tr>
<tr>
<td>BR11</td>
<td>45#</td>
<td>47</td>
</tr>
<tr>
<td>BR13</td>
<td>49#</td>
<td>55</td>
</tr>
<tr>
<td>BPI4</td>
<td>50#</td>
<td>51</td>
</tr>
<tr>
<td>BPI6</td>
<td>28</td>
<td>42#</td>
</tr>
<tr>
<td>BU1</td>
<td>201#</td>
<td>207</td>
</tr>
<tr>
<td>BU2</td>
<td>203#</td>
<td>204</td>
</tr>
<tr>
<td>BURST</td>
<td>165</td>
<td>159</td>
</tr>
<tr>
<td>CI1</td>
<td>121#</td>
<td>123</td>
</tr>
<tr>
<td>CI2</td>
<td>125#</td>
<td>126</td>
</tr>
<tr>
<td>CI3</td>
<td>129#</td>
<td>142</td>
</tr>
<tr>
<td>CI4</td>
<td>130#</td>
<td>131</td>
</tr>
<tr>
<td>CI5</td>
<td>137</td>
<td>143</td>
</tr>
<tr>
<td>CIN</td>
<td>30</td>
<td>118</td>
</tr>
<tr>
<td>CKRATE</td>
<td>155#</td>
<td>248</td>
</tr>
<tr>
<td>C01</td>
<td>97#</td>
<td>110</td>
</tr>
<tr>
<td>C02</td>
<td>101#</td>
<td>182</td>
</tr>
<tr>
<td>COUT</td>
<td>35</td>
<td>81</td>
</tr>
<tr>
<td>CRT1</td>
<td>26#</td>
<td>33</td>
</tr>
<tr>
<td>CRTTST</td>
<td>25#</td>
<td></td>
</tr>
<tr>
<td>CYCNO</td>
<td>153#</td>
<td>200</td>
</tr>
<tr>
<td>ECHI</td>
<td>30#</td>
<td>36</td>
</tr>
<tr>
<td>HALFR1</td>
<td>14#</td>
<td>71</td>
</tr>
<tr>
<td>HALFCY</td>
<td>154#</td>
<td>202</td>
</tr>
<tr>
<td>LOR</td>
<td>157#</td>
<td>214</td>
</tr>
<tr>
<td>LEADER</td>
<td>156#</td>
<td>163</td>
</tr>
<tr>
<td>F81</td>
<td>215#</td>
<td>218</td>
</tr>
<tr>
<td>F82</td>
<td>219#</td>
<td>222</td>
</tr>
<tr>
<td>PLAYB</td>
<td>214#</td>
<td>216</td>
</tr>
<tr>
<td>S1</td>
<td>77#</td>
<td>83</td>
</tr>
<tr>
<td>STGON</td>
<td>29</td>
<td>76#</td>
</tr>
<tr>
<td>STRNG</td>
<td>76#</td>
<td>85#</td>
</tr>
<tr>
<td>TAPEIN</td>
<td>219#</td>
<td>227#</td>
</tr>
<tr>
<td>TAPEO</td>
<td>171</td>
<td>179#</td>
</tr>
<tr>
<td>TI1</td>
<td>228#</td>
<td>245</td>
</tr>
<tr>
<td>TI2</td>
<td>229#</td>
<td>231</td>
</tr>
<tr>
<td>TI3</td>
<td>234#</td>
<td>236</td>
</tr>
<tr>
<td>TO1</td>
<td>182#</td>
<td>195</td>
</tr>
</tbody>
</table>

CROSS REFERENCE COMPLETE
Related Intel Publications

"8085 Microcomputer Systems User's Manual"

"8080/8085 Assembly Language Programming Manual"

The material in this Application Note is for informational purposes only and is subject to change without notice. Intel Corporation has made an effort to verify that the material in this document is correct. However, Intel Corporation does not assume any responsibility for errors that may appear in this document.

The following are trademarks of Intel Corporation and may be used only to describe Intel Products:

MCS
SDK-85
ICE-85