80C451 operation of port 6 AN408

INTRODUCTION

The features of the 80C451 are shared with the 80C51 or are conventional except for the operation of port 6. The flexibility of this port facilitates high-speed parallel data communications. This application note discusses the use of port 6 and is divided into the following sections:

1. Port 6 as a processor bus interface.
2. Using port 6 as a standard pseudo bidirectional I/O port.
3. Implementation of parallel printer ports.

This information applies to all versions of the part: 80C451, 83C451, and the 87C451.

PORT 6 AS A PROCESSOR BUS INTERFACE

Port 6 allows use of the 80C451 as an element on a microprocessor type bus. The host processor could be a general purpose MPU or the data bus of a microcontroller like the 80C451 itself. This feature allows single or multiple 80C451 controllers to be used on a bus as flexible peripheral processing elements. Applications could include keyboard scanners, serial I/O controllers, servo controllers, etc.

OPERATION

On reset, port 6 is programmed correctly for use as a bus interface (see 2). This prevents the interface from disrupting data on the bus of the host processor during power-up. Software initialization of the CSR (Control Status Register) is not required. A dummy read of port 6 may be required to clear the IBF (Input Buffer Full) flag since it could be set by turn on transients on the bus of the host processor. On reset, the CSR of the 83C451 is programmed to allow the following:

1. AFLAG is an input controlling the port select function. If AFLAG is high, the contents of the CSR is output on port 6 when the port is read by the host. If AFLAG is low, then the contents of the output latch is output when port 6 is read by the host.
2. BFLAG is an input controlling the port enable function. In this mode when BFLAG is high, the input latch and the output drivers are disabled and the flags are not affected by the IDS (Input Data Strobe) or ODS (Output Data Strobe) signals. When BFLAG is low, the port is enabled for reading and writing under the control of IDS and ODS pins.

Figure 1 shows one possible example of an 80C451 on a memory bus. This arrangement allows the main processor to query port 6 for flag status without interrupting the 80C451. If the address decoder, shown in Figure 1, enables port 6 on the 80C451 when the address is 8000H or 8001H, and the address line A0 controls the port select feature, then the host processor can read and write to port 6 using address 8000H. Since the port select function is being controlled by the address line A0, the CSR contents can be read by the host processor at address 8001H.

By testing the CSR contents in this way, the host processor can tell if new data has been written to the port 6 output latch since it last read the port or if the 80C451 has read the last byte that the host wrote to the port. Conversely, the 80C451 can poll the flags in its CSR to see if the host processor has written to or read from port 6 since the last time it serviced the port.

If desired, an interrupt source for the 80C451 can be derived easily from the port enable source as shown by the dashed line in Figure 1.

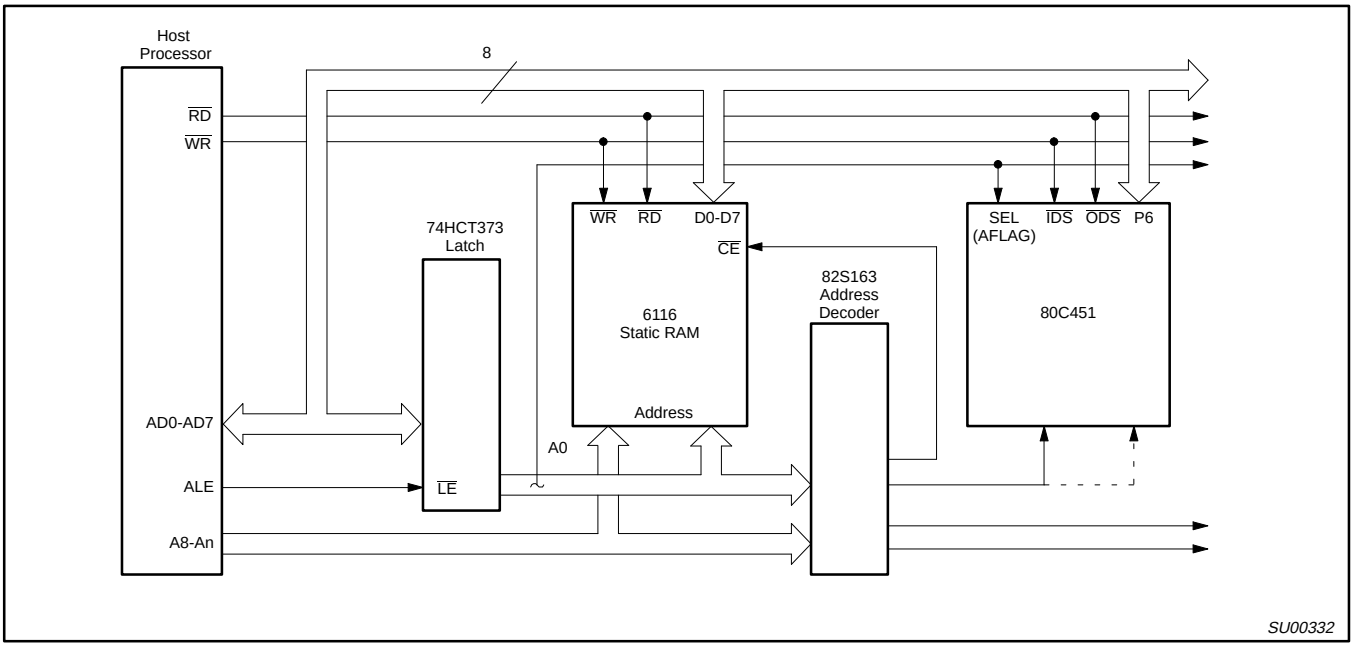

Figure 1. An 83C451 on a Microprocessor Memory Bus

SOFTWARE EXAMPLES

To write to port 6 on the bus shown in Figure 1, the host processor first reads the CSR contents at address 8001H, and tests the input buffer full flag (CSR bit 0). If the flag is clear, the host writes a byte to address 8000H. This loads the input buffer latch of port 6 and sets the input buffer full flag. Conversely, the 80C451 polls the IBF flag and reads a byte from port 6 when it finds the flag set. The flag is automatically reset when this internal read occurs.

80C451 ROUTINE TO READ ONE BYTE FROM HOST VIA PORT 6

```
RCVR:   JNB CSR.0,RCVR   ;TEST IBF FLAG
        MOV A,P6         ;WHEN FLAG IS SET READ BYTE
        RET
```

80C451 ROUTINE TO WRITE ONE BYTE TO THE 83C451 PORT 6

If the host processor is an 80C51, the following routine will write a byte of data to the 80C451. The data involved is passed to the routine through register 1.

```
XMIT:   MOV DPTR,8001H
TEST:   MOVX A,@DPTR     ;READ THE CSR
        JB ACC.0,TEST    ;TEST IBF FLAG
        MOV DPTR,8000H
        MOV A,R1
        MOVX @DPTR,A     ;WRITE DATA TO THE 451
        RET
```

80C451 ROUTINE TO WRITE ONE BYTE TO HOST VIA PORT 6

Routines for data transfer in the opposite direction are similar to the above two. The 80C451 version is given below.

```
XMIT:   JB CSR.1,XMIT    ;TEST OBF FLAG
        MOV P6,A         ;WRITE DATA
        RET
```

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
1	1	1	1	1	1		

SU00333

Figure 2. CSR Programmed to Allow Port 6 as a Bus Interface

USING PORT 6 AS A STANDARD QUASI-BIDIRECTIONAL I/O PORT

To use port 6 as a common I/O port, all of the control pins are tied to ground (see Figure 3). On hardware reset, bits 2 - 7 in the CSR are set to one. Port operation and electrical characteristics become identical to port 1 on the 80C51 and the 80C451 ports 1, 4, and 5. No software initialization is required.

If desired, AFLAG and BFLAG can be used as outputs while port 6 is operating as a standard quasi-bidirectional I/O port (see Figure 4). In this case, only $\overline{IDS}$ and $\overline{ODS}$ are tied to ground and the CSR is initialized to allow operation of AFLAG and BFLAG as simple outputs (see Figure 5).

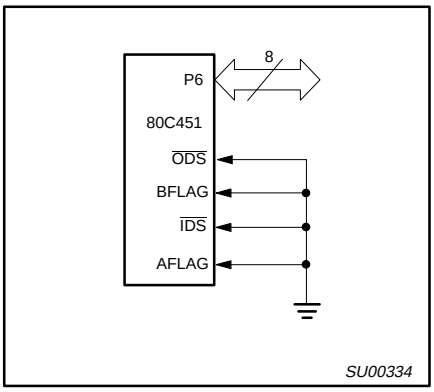

Figure 3. Standard I/O Port on Reset

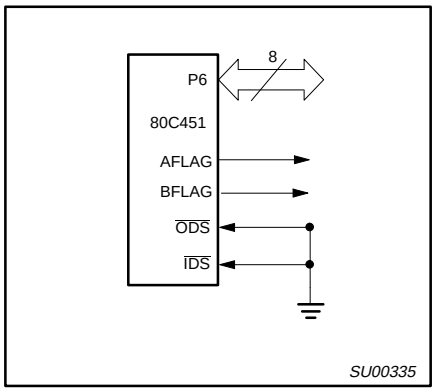

Figure 4. Standard I/O Port on Reset with AFLAG and BFLAG as Outputs

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
1	X	0	X	X	1		

SU00336

Figure 5. CSR Programmed to Allow AFLAG and BFLAG to Operate as Outputs and Port 6 as a Standard I/O Port

DATA TRANSFER SIGNAL PINS		
Pin No.	Groung Return Pin No.	Signal
1	19	$\overline{STROBE}$
2	20	DATA 1
3	21	DATA 2
4	22	DATA 3
5	23	DATA 4
6	24	DATA 5
7	25	DATA 6
8	26	DATA 7
9	27	DATA 8
10	28	$\overline{ACKNLG}$
11	29	BUSY

TYPICAL AUXILIARY PIN FUNCTIONS	
Pin No.	Signal
12	PAPER OUT
14	$\overline{AUTO\ LINE\ FEED}$
16	LOGIC GROUND
17	CHASSIS GND
30	GROUND RETURN
31	$\overline{RESET\ PRINTER}$
32	$\overline{ERROR}$
33	GROUND RETURN
36	$\overline{SLCT\ IN}$

SU00337

Figure 6. Parallel Printer Interface Pin Functions

IMPLEMENTATION OF PARALLEL PRINTER PORTS USING PORT 6

The 80C451 is an excellent choice for a printer controller. The 80C451 has the facilities to permit all of the intelligent features of a common printer to be handled by a single chip:

1. The features of port 6 allow a parallel printer port to be designed with only line driving and receiving chips required as additional hardware.
2. The onboard UART allows RS232 interfacing with only level shifting chips added.
3. The 8-bit parallel ports 0 to 6 are ample to drive onboard control functions, even when ports are used for external memory access, interrupts, and other functions.
4. The RAM addressing ability of ports 0 and 2 can be used to address up to 64k bytes of a hardware buffer/spooler. AFLAG and BFLAG as simple outputs (see Figure 5).
5. The 64k byte ROM addressing capability allows space for the most sophisticated software.

In addition, either end of a parallel interface can be implemented using port 6, and the interfaces can be interrupt driven or polled in either case.

THE INTERFACE

Data transfer on a parallel printer interface occurs across eleven signal lines. The other conductors on the standard plug are used as ground returns or for auxiliary functions (see Figure 6). Only the data transfer signals will be considered.

The Data Transfer Format

The parallel printer interfaces are far more standardized in features than their serial counterpart. However, at least three significant variations exist in handshake style in printers using generic parallel interfaces. This fact influences the design of both port hardware and software. A good transmitter should be able to drive devices with all three styles of handshakes, and a good receiver should generate the handshake most likely compatible with any transmitter.

The Variations

Type 1—Figure 7 shows a common style of handshake and is the style that will be implemented in the receiver examples. A busy signal and an acknowledge strobe pulse are generated for every byte received.

Type 2—Another style of handshake generates a busy signal only when the printer will not be able to accept more data for a relatively long time. Acknowledge pulses are created after every byte received. When the busy signal is generated after a byte is received, the associated acknowledge pulse does not occur until *after* the busy signal returns to logic zero (see Figure 7).

Type 3—A third handshake style does not generate acknowledge pulses, but a busy signal is produced after every byte is received.

PARALLEL PRINTER INTERFACES USING POLLING

Transmitter Operation

This application illustrates the flexibility of the port 6 logic in solving an applications problem. We need to be able to handle all types of acknowledge signals that might be received by the transmitter. We will use the $\overline{\text{ODS}}$ pin and output buffer full flag logic to record the receipt of the acknowledge pulse (see Figure 8), but not all parallel receivers generate acknowledge pulses. We could poll the busy signal line, but not all receivers generate busy signals for each byte received; so lack of a busy signal does not imply that we can send another byte. We can, however, expect an acknowledge pulse very shortly after the end of a busy signal if one is going to arrive at all. So we can send a new data byte after having received either a positive transition on the acknowledge line, or shortly after receiving a negative edge on the busy line.

The CSR is programmed to the output only mode. In this mode, the $\overline{\text{ODS}}$ pin does not control the output drivers but only the output buffer full flag. The flag serves to record the positive transition of the acknowledge signal. The input latch is not used, but the $\overline{\text{IDS}}$ pin is used to set the input buffer full flag. This is used to record the negative transition at the end of the busy signal. Dummy reads by the 80C451 of port 6 will be used to clear the flag. In this example, the AFLAG mode is set only to place the port in the output only mode. The AFLAG pin is not actually used (see Figure 10).

The transmitter's CSR (control status register) is programmed to the following mode (see Figure 9):

1. CSR bit 6 controls the BFLAG output and therefore the strobe line.
2. The OBF (output buffer full) flag controls the AFLAG output.
3. The OBF is cleared on the positive edge of the ODS input.
4. The IBF flag is cleared on the negative edge of the IDS strobe.

NOTE:
With this combination of modes set, port 6 is in the output only mode.

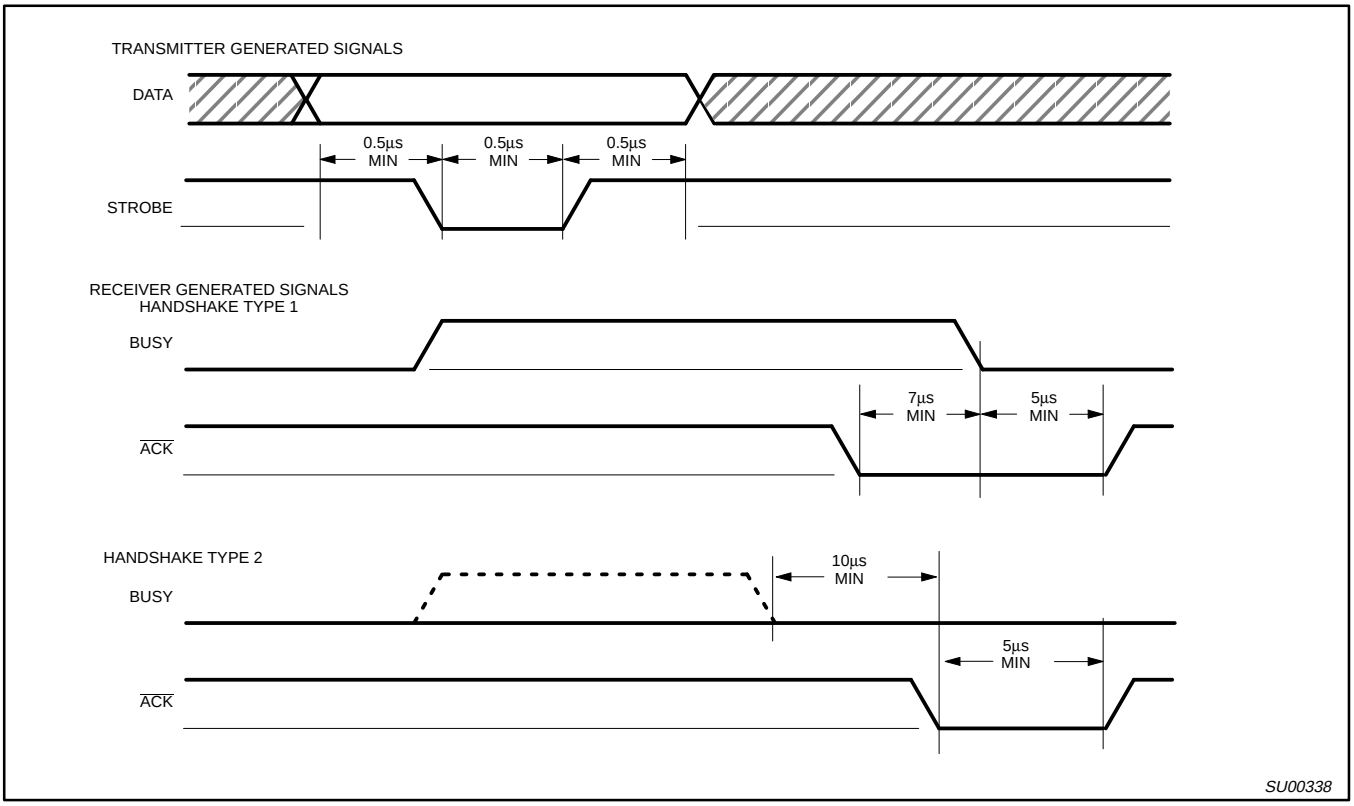

Figure 7. Parallel Printer Interface Signals

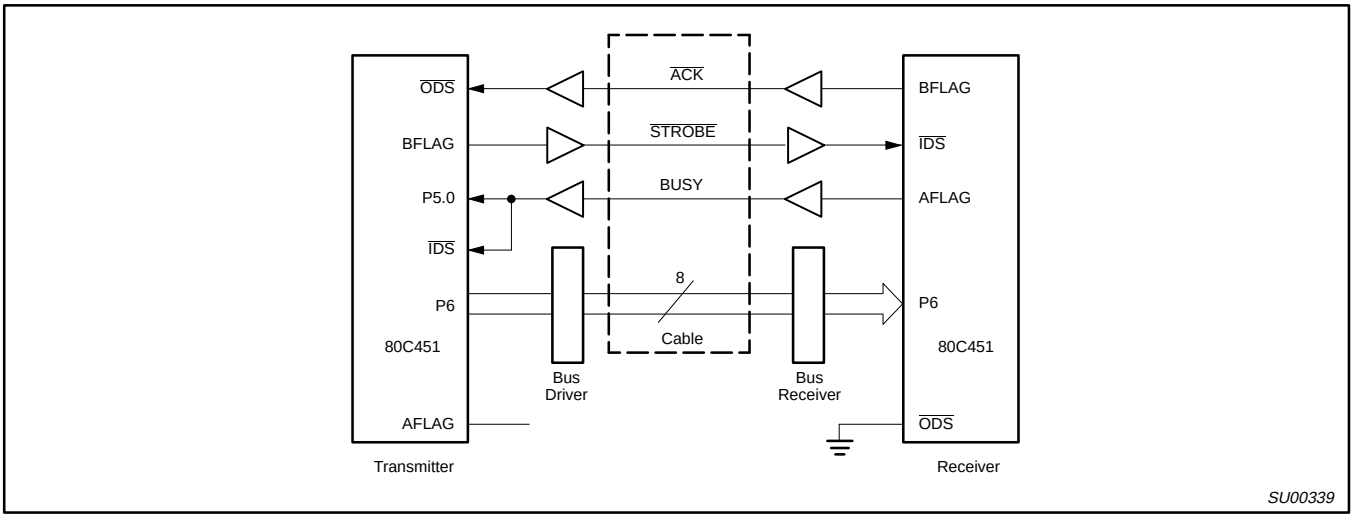

Figure 8. Interconnection for a Parallel Interface Using Polling

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
0	1	1	0	0	1		

SU00340

Figure 9. CSR Programmed for Polled Transmitter Operation

Receiver Operation

In receiver operation, the IDS input is used to latch in the data transmitted on receipt of the strobe pulse. The receiver's CSR is programmed to allow the following (see Figure 11):

1. The input buffer full flag is output through the BFLAG pin and is used as the busy signal to the transmitter.
2. The IBF flag is set and data is latched on the positive edge of IDS.
3. Writing to the CSR bit 4 controls the AFLAG output and therefore the acknowledge line.

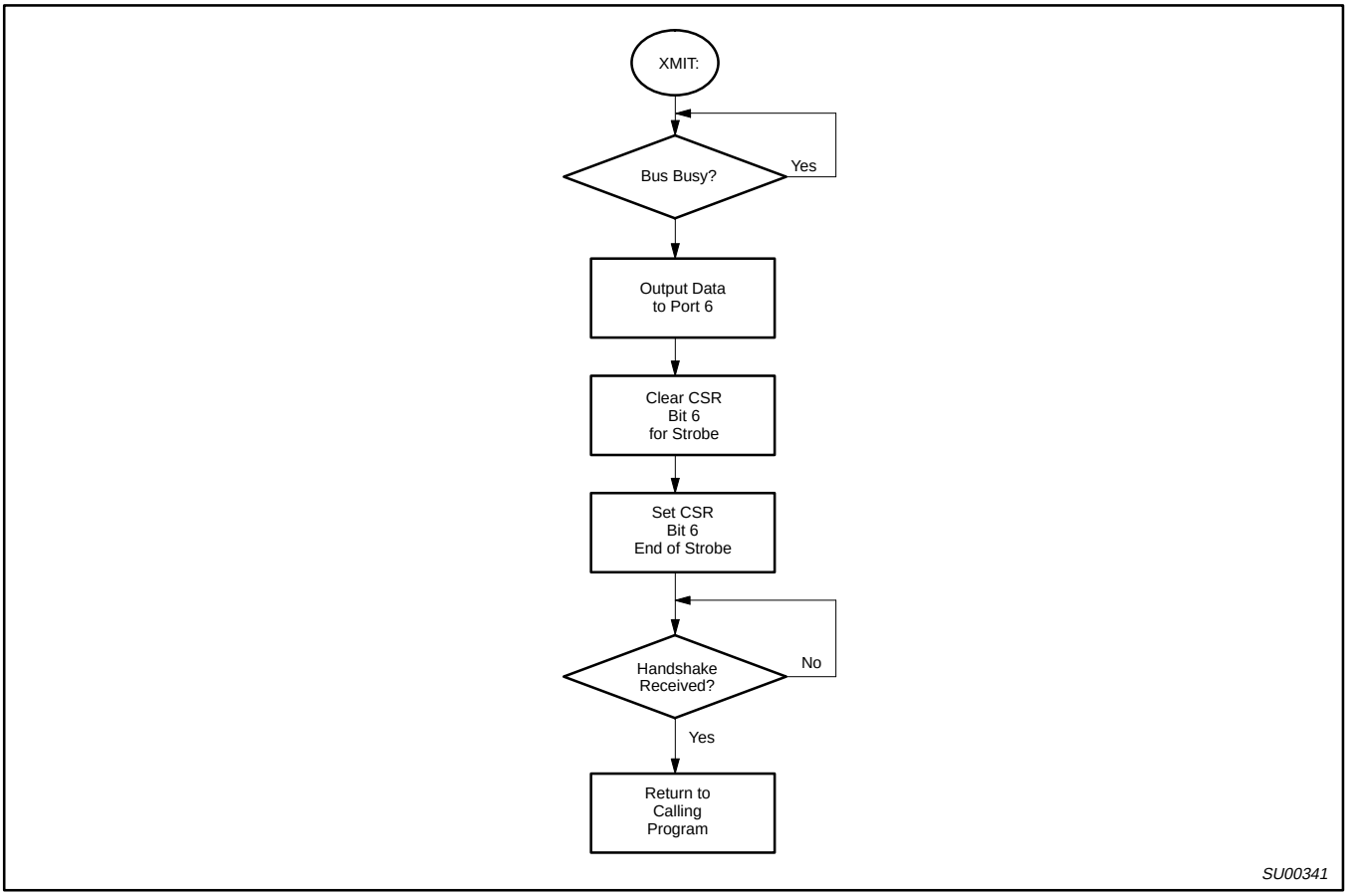

Figure 10. Flow Chart of Polled Parallel Transmitter Operation

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
1	0	0	1	1	0		

SU00342

Figure 11. CSR Programmed for Polled Parallel Receiver Operation

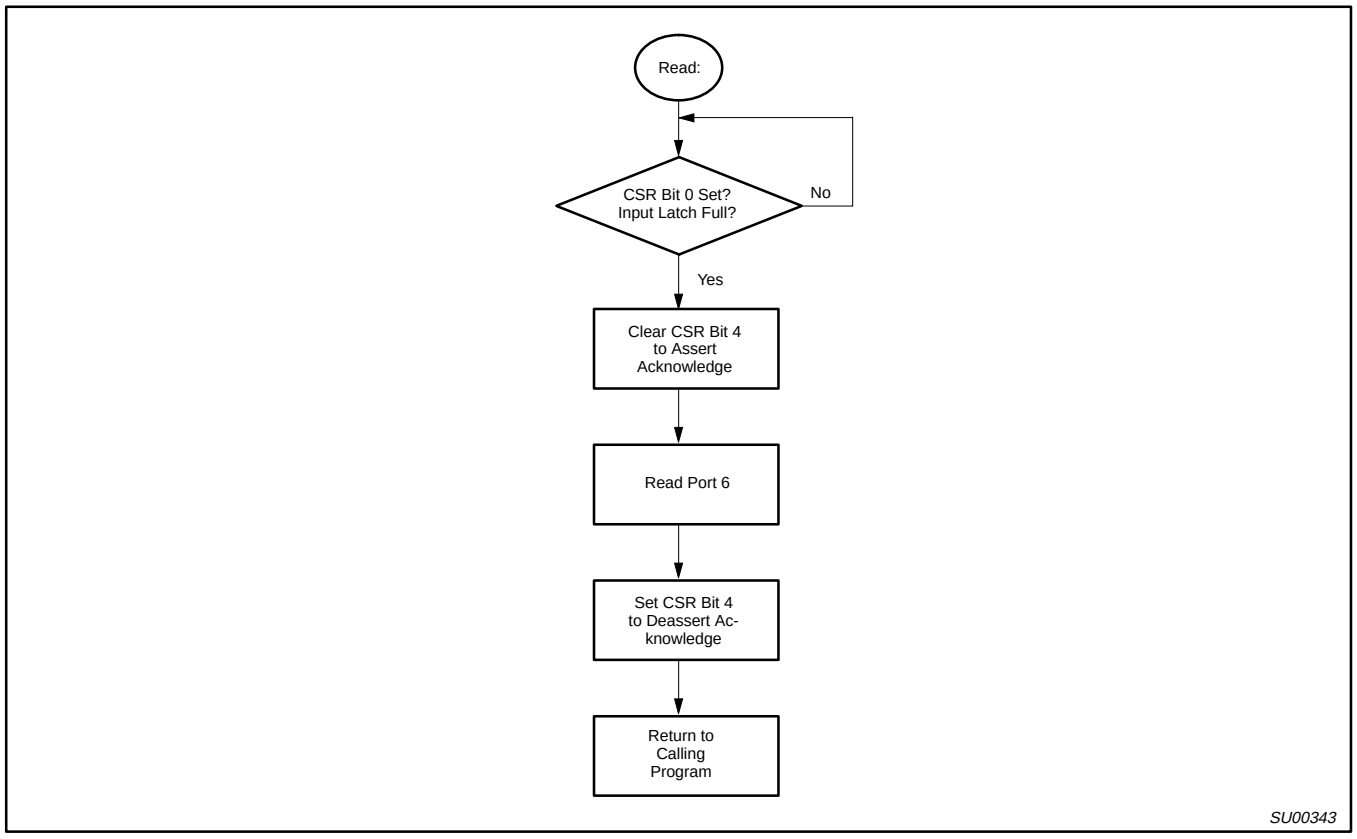

Figure 12. Flow Chart of Polled Parallel Receiver Operation

SOFTWARE EXAMPLES

This polled parallel transmit routine outputs one byte passed to it in the accumulator.

```
P_INIT:   MOV CSR,#064H     ;INITIALIZE PORT 6 OPERATING MODE
P_OUT:    JB P5.0           ;WAIT IF BUSY SIGNAL IS HIGH
          MOV P6,ACC        ;OUTPUT DATA
          MOV R1,P6         ;DUMMY READ TO CLEAR IBF FLAG
          MOV R1,#02H       ;INITIALIZE DELAY COUNTER
          CLEAR CSR.6       ;START STROBE PULSE
          DJNZ R1,$         ;TIME 6 MICROSECOND STROBE PULSE
          SETB CSR.6        ;END STROBE PULSE
WAIT:     JNB CSR.1,OUT     ;EXIT IF ACKNOWLEDGE RCV'D
          JNB CSR.0,WAIT    ;EXIT IF NEGATIVE BUSY EDGE RCV'D
```

This polled parallel receive routine places one byte in the accumulator each time it is called.

```
P_INIT:   MOV CSR,#09CH     ;INITIALIZE PORT 6 OPERATING MODE
          MOV R7,P6         ;DUMMY READ TO CLEAR IBF FLAG
P_IN      JNB CSR.0         ;INPUT BUFFER LATCH FULL?
          CLR CSR.4         ;BEGIN ACKNOWLEDGE PULSE
          MOV R7,#02H       ;INITIALIZE DELAY COUNTER
          DJNZ R7,$         ;TIME ACKNOWLEDGE PULSE
          MOV A,P6          ;READ BYTE – CLEAR BUSY SIGNAL
          MOV R7,#02H       ;INITIALIZE DELAY COUNTER
          DJNZ R7,$         ;TIME ACKNOWLEDGE PULSE
          SETB CSR.4        ;END ACKNOWLEDGE PULSE
          RET
```

INTERRUPT DRIVEN PARALLEL PRINTER INTERFACE

(See Figure 13)

Transmitter Operation

The transmitter's CSR (control status register) is programmed to the following mode (see Figure 14):

1. CSR bit 6 controls the BFLAG output and therefore the strobe line.
2. The OBF (output buffer full) flag controls the AFLAG output.
3. The OBF is cleared on the positive edge of the $\overline{ODS}$ (output data strobe) input.
4. The IBF flag is set on the negative edge of the $\overline{IDS}$ (input data strobe) pin.

NOTE:
With this combination of AFLAG and BFLAG modes set, port 6 is in the output only mode. The output drivers are always enabled and the $\overline{ODS}$ input is only used to clear the OBF flag.

INTO is programmed to be negative edge sensitive and is connected to the OBF flag through the AFLAG pin. The OBF is cleared on the positive edge of $\overline{ODS}$. The net result is that INTO is triggered on the end of the ACK pulse (a positive edge). This signals the transmitter that another byte may be transmitted. The transmitting 83C451 is free to do other tasks prior to this interrupt.

In this routine, Figure 15, the main program establishes a buffer in data memory ended by an ASCII end of text character. To begin outputting the buffer, the routine PSEND is called. The rest of the buffer is emptied by the interrupt vectors to PSEND1.

For printers which generate acknowledge pulses, output rates of 25k transfers per second are achieved. Timer generated interrupts are used to periodically return program execution to the routine to service non-acknowledging printers and to provide a timeout feature. Non-acknowledging printers are serviced at a rate of about 2.5k transfers per second. This maximum rate may be varied by adjusting the timer reload value. As written, the time out procedure attempts to retransmit a byte when the printer has not acknowledged for an excessively long time.

Receiver Operation

In receiver operation, the IDS input is used to latch in the data transmitted on receipt of the strobe pulse. The receiver's CSR is programmed to allow the following (see Figure 16):

1. The input buffer full flag is output through the BFLAG pin and is used as the busy signal to the transmitter. The IBF flag is set and data is latched on the positive edge of $\overline{IDS}$.
2. Writing to the CSR bit 4 controls the AFLAG output and therefore the acknowledge line.

The receiver is interrupted on the negative edge of the data strobe. Data is latched in on the positive edge of the strobe pulse (see Figure 17). Since the strobe pulse is normally very short, there is little time lost between receiving the interrupt and having valid data in the input latch. The receiver is free to do other tasks prior to receiving the $\overline{INTO}$ interrupt.

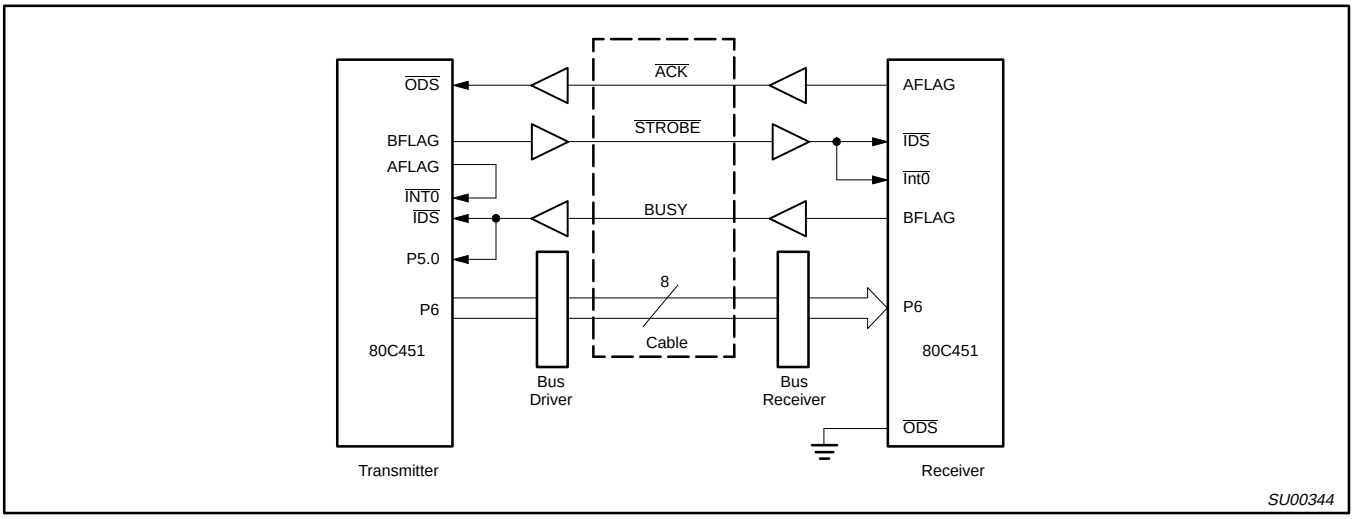

Figure 13. Interrupt Driven Parallel Interfaces Using 80C451 Controllers

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
0	1	1	0	1	1		

SU00345

Figure 14. CSR Programmed for Use as an Interrupt Driven Parallel Transmitter

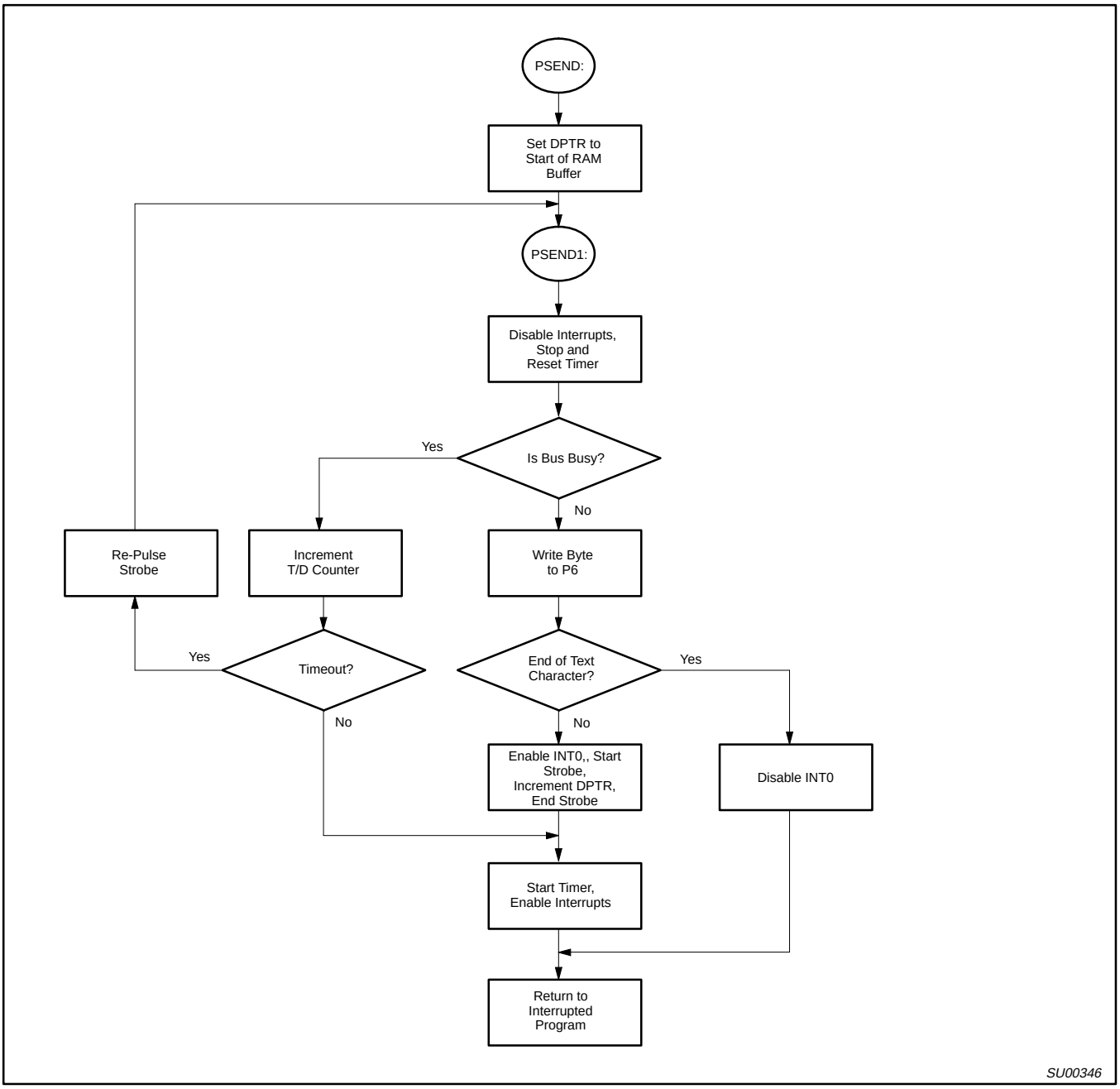

Figure 15. Flow Chart for an Interrupt Driven Parallel Transmitter

CSR 7	CSR 6	CSR 5	CSR 4	CSR 3	CSR 2	CSR 1	CSR 0
MB1	MB0	MA1	MA0	OBFC	IDSM	OBF	IBF
1	0	0	1	1	0		

SU00347

Figure 16. CSR Programmed for Use as an Interrupt Driven Parallel Receiver

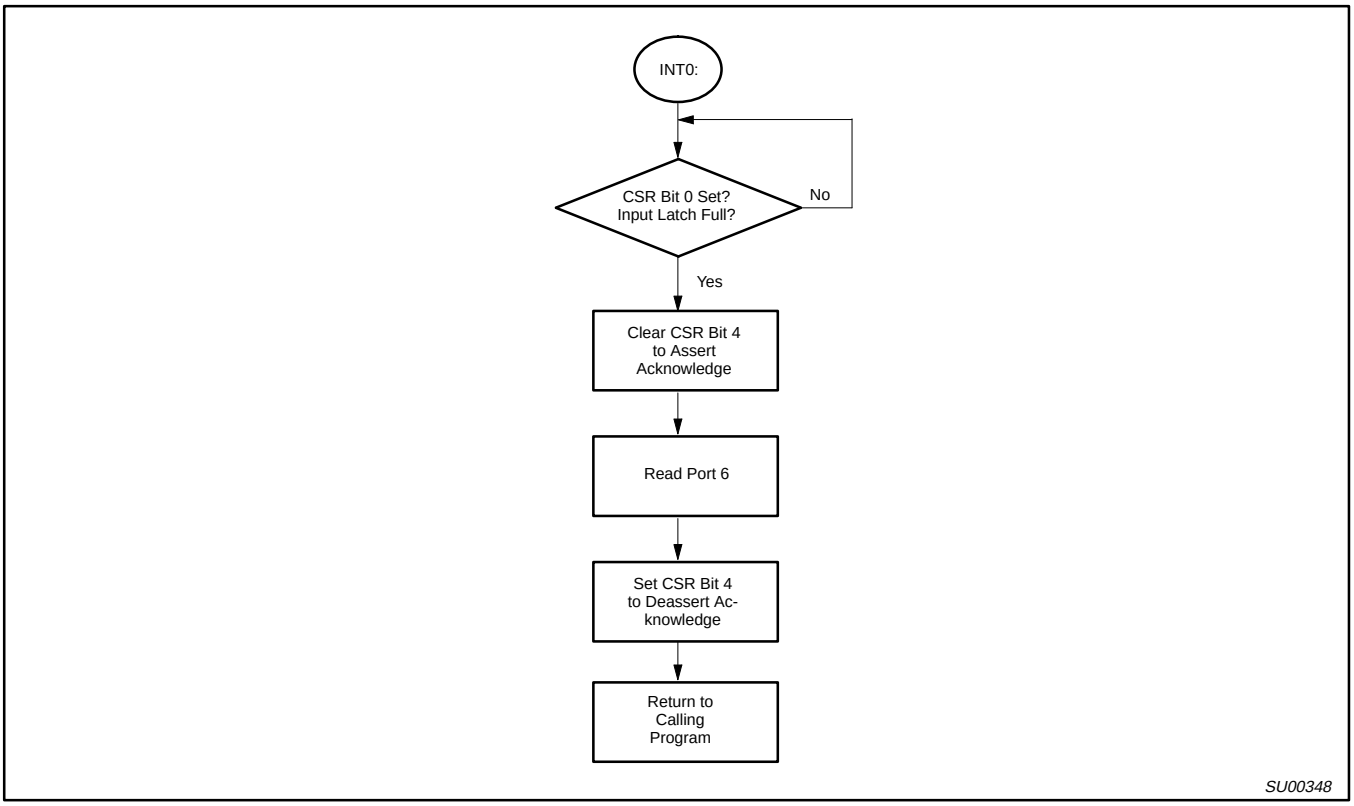

Figure 17. Flow Chart of Interrupt Driven Parallel Receiver Operation

SOFTWARE EXAMPLES

The software for the interrupt driven parallel receiver is similar to the polled receiver example. However, after an interrupt is received, this routine checks to confirm that data has been latched by the positive edge of the strobe pulse before proceeding with the routine.

```
    INIT:       MOV CSR,#090H       ;INITIALIZE CSR
                SETB EX0            ;ENABLE INTERRUPT 0
                SETB IT0            ;SET NEG EDGE TRIGGERED INTERRUPTS
                SETB EA             ;ENABLE ALL INTERRUPTS
    ORG EXTI0                       ;INTERRUPT 0 VECTOR
                JMP RCVR

                RCVR:
RCVR:           JNB CSR.0,#         ;CONFIRM DATA LATCHED
                CLR CSR.4           ;START ACKNOWLEDGE PULSE
                MOV R7,#02H         ;INITIALIZE THE DELAY COUNTER
                DJNZ R7,#           ;TIME ACK PULSE
                MOV A,P6            ;READ BYTE – RESET BUSY LINE
                MOV R7,#02H         ;INITIALIZE THE DELAY COUNTER
                DJNZ R7,$           ;TIME ACK PULSE
                SETB CSR.4          ;END ACK PULSE
                RET1
```

This is the software for the interrupt driven parallel transmitter example.

```
; XMIT ROUTINE DRIVEN BY ACK PULSE GENERATED INTERRUPTS, OR TIME GENERATED INTERRUPTS
; FOR NON ACKNOWLEDGING PRINTERS. READS DATA BUFFER IN EXTERNAL RAM STARTING AT 100H
; AND READING UNTIL 04H IS FOUND.

        ORG   RESET
         JMP  26H
        ORG   TIMER0
         JMP  PSEND1
        ORG   EXTI0
         JMP  PSEND1
        ORG   26H
              MOV CSR,#064H          ;PORT 6 MODE
              MOV TMOD,#002H         ;CONFIGURE TIMER 0 TO 16 BITS
              SETB TO0               ;INT0 IS EDGE TRIGGERED
              SETB EA                ;ENABLE INTERRUPTS
PSEND:        MOV DPTR,#0100H        ;SET DPTR TO START OF TEXT
                                     ;BUFFER
PSEND1:       CLR EA                 ;DISABLE INTERRUPTS AND STOP
                                     ;TIMER
              CLR TR0                ;IF ENABLED
              CLR ET0
              MOV R7,00H             ;CLEAR TIMEOUT COUNTER
              MOV R6,00H
              MOV TH0,#–4            ;SET TIMER INTERRUPT PERIOD
              MOV TL0,#00H
              JB 0C8H,BB             ;BUS BUSY
              MOV ACC,#00H           ;CLEAR ACCUMULATOR
              MOVX 1,@DPTR           ;RETRIEVE FIRST BYTE
              MOV 06,ACC             ;OUTPUT FIRST BYTE
              CJNE A,#004H, CONT1    ;LOOK FOR END OF TEXT
              JMP EOTB
CONT1:        SETB ERX0              ;ENABLE INTO
              CKR 0EEH               ;START STROBE PULSE
              INC DPTR
              MOV ACC,DPH            ;LOOK FOR PHYSICAL END OF
              JB ACC.2,EOTB          ;TEXT BUFFER
              SETB 0EEH
              JMP CONT
EOTB:         CLR EX0                ;END OF TEXT FOUND, DISABLE
                                     ;INT0
              SETB 0EEH
              SETB EA
              RETI
BB:           INC R7                 ;COUNT TIMER TIMEOUTS ON
                                     ;BUS BUSY
              CJNE R7,#00H, CONT     ;LOOK FOR OVERFLOW
              INC R6                 ;COUNT OVERFLOWS
              CJNE R6,#10H, CONT     ;TIMEOUT APPROX 5 SEC
              JMP TO
CONT:         SETB TR0               ;ENABLE TIMER INTERRUPT
              SETB ET0               ;START TIMER
              SETB EA
              RETI
TO:           CLR 0C9H               ;SEND NEW STROBE PULSE IN
                                     ;RESPONSE TO TIMEOUT
              NOP
              NOP
              MOV R6,#00H            ;RESET TO COUNTER
              MOV R7,#00H
              SETB 0C9H              ;END OF STROBE PULSE
              JMP PSEND1
```