256k Centronics printer buffer using the 87C451 microcontroller

AN417

DESCRIPTION

This application note describes a stand alone Centronics type parallel printer buffer using the 87C451 expanded I/O microcontroller. This type of unit would typically be placed between a personal computer and its printer. It captures the data to be printed at high speed, freeing the personal computer to go to other tasks, and sends data to the printer as required. As described here, 256k dynamic RAMs are used, providing over one quarter million characters of storage. If desired the design is easily modified to work with 1 megabit DRAMs. Although written with the 87C451 in mind, this design is applicable to the 80C451 and 83C451.

Design Objectives

The objectives kept in mind during the design of this device were: provide a substantial size of buffer, keep the parts count and the power consumption to a minimum, and use readily available components.

A buffer size of 256k bytes was chosen because, although a 64k byte buffer is very easily implemented using the 8051 family's 64k external data storage capabilities, it is a little too small for today's printing applications that print a page of text in graphics mode, using up twenty times as many bytes as standard printing mode. Presenting a method for controlling 256k DRAMs shows off the I/O capabilities of the 87C451, and it is very easy to add the extra address line for one megabit devices if a larger buffer is needed.

The 8XC451 Microcontroller

The 8XC451 is an 8-bit microcontroller based on the familiar 8051 family of devices. In fact, it is an 80C51 with three added ports: P4, P5, and P6. Ports 4 and 5 give 12 (16 in PLCC) additional quasi-bidirectional I/O lines. Port 6 provides another 8 bits of I/O, plus 4 handshake lines that can be programmed to operate in several useful modes for interfacing. The 8XC451 comes in three versions: ROMless 80C451, 83C451 with $4k \times 8$ ROM, and 87C451 with $4k \times 8$ EPROM.

In this note, port 6 is used in the I/O mode as a Centronics compatible printer output port. Additionally, the /IDS and BFLAG pins normally associated with port 6 are used as part of the input port logic. For a complete discussion of port 6 operating modes and programming, see the application note AN408 titled "83C451 Microcontroller Operation of Port 6."

Circuit Description

Figure 1 is a schematic diagram of the printer buffer circuit. Other than the 87C451 (U1), and the eight 256k DRAMs (U5-U12), only two 74LS244 buffers (U2, U3) and a 76HCT374 (U4) octal flip-flop are needed. The U2 and U3 buffers are included to provide full drive capability for the output port and some of the handshake signals on the input port, as the output buffers on the 87C451 can only drive 3 LSTTL loads. U4 has 8-bit data strobed into it by the /STB pulse of the input port.

As the code size for this application is quite small (less than 1k bytes), the on-chip instruction memory is quite sufficient for program storage. For a production version, the 87C451 could be replaced with the 83C451 with a $4k \times 8$ masked ROM on chip. Note that port 0 and port 1 are not used in the present design; thus the 80C451 may be used in this application with the addition of an external address latch and EPROM.

The /RAS, /CAS, and /WR signals for the DRAM array are provided by port 3 bits /WR, /RD, and T1. Note that as in the 80C51, all port 3 signals are multifunctional. That is, each can be treated as a regular quasi-bidirectional port bit, or as having the special function indicated by its name. This feature is an advantage when using /WR and /RD as /RAS and /CAS control signals for a DRAM array. Treated as a normal port bit, the /WR pin is cleared and set by individual CLR and SETB instructions for a normal length RAM read or write cycle. However, when performing a refresh cycle, /RAS (port 3/WR) can be pulsed low using a dummy MOVX @R0,A (move to external data memory) instruction. This allows DRAM refresh to be done much more quickly than would otherwise be possible.

Port 1 and one bit from port 4 form the 9-bit address required when addressing the DRAM array. The data inputs to the array come from the parallel input data lines which are latched by U4. The RAM data outputs are fed to port 5. By making the data outputs available to the processor, it is possible to add some additional features to the firmware, such as control codes for printing multiple copies of a document, data compression, data conversion, etc. which are not implemented in this design.

Port 6 Operation

The /IDS (input data strobe) and BFLAG pins are normally used in conjunction with the port 6 bidirectional mode. In this mode, the /IDS pin is used to strobe data into the port 6 input latches, and BFLAG is used as flag output. In this application, however, these two bits are used to good effect as part of the (separate) input port logic. When a byte of data is strobed into U4 by the printer port of the host computer, the /STB signal connected to /IDS sets the input buffer full flag (IBF). BFLAG is programmed to mirror the contents of IBF, and therefore becomes asserted. This makes it ideal to be used as the BUSY output for the input port. After the input port data has been read and stored in the RAM buffer, BFLAG is de-asserted by performing a dummy read of port 6, which clears IBF. To complete the input port logic, one of the port 3 pins, P3.4, is used as the acknowledge signal, and is asserted/de-asserted by software. The /ODS pin is tied to ground to permanently enable the port 6 output drivers. This does not cause difficulty as no data is being input into the port.

Note that programming port 6 to operate in the bidirectional mode as described above means the loss of /ODS as an acknowledge input. The acknowledge input is normally used to clear the OBF (output buffer full) flag, indicating that the printer is ready for another character. On the other hand, operating port 6 in the "output only" mode causes the loss of BFLAG as BUSY output. Because the input port requires an instant BUSY indication while the output port only needs to remember the occurrence of an acknowledge pulse, it makes sense to program port 6 to operate in the bidirectional mode, with /ODS grounded to enable the output drivers. The /INT1 pin can be used instead of /ODS to record the occurrence of an acknowledge pulse with the interrupt system.

Priority and Execution of Tasks

There are three tasks that must be performed in this system: Receive—servicing the input port and storing the input character; Transmit—sending stored characters to the output port as required; and Refresh—performing DRAM refresh. The timers and interrupt system are used to manage the execution and priority of these tasks. Figure 2 and Figure 3 illustrate the flow charts of these tasks. Firmware, broken into sections, performing these three functions as well as an initialization routine is provided.

The 51C256 DRAMs require a 256 row refresh every 4 milliseconds. Rather than do an entire refresh cycle every 4 milliseconds, it is done as 64 rows every millisecond. This leaves time for other tasks to get service "slices" more frequently. As DRAM refresh is obviously the highest priority, timer 0 is used as the refresh interval timer, and is programmed to the 16-bit mode, and set to the higher priority level in the interrupt priority (IP) register. The refresh code is written in-line rather than in a loop to maximize speed.

An interesting point to note is that when there are no characters stored, the DRAM does not need to be refreshed. If power consumption

is of concern, the 87C451 could be programmed to go into idle mode whenever the buffer were empty. A character strobed into the input port would cause an interrupt, restarting the 87C451; DRAM refresh would be maintained until the buffer was once again empty.

The next highest priority should be input port service, as the reason for having a printer buffer is to get the data out of the computer as quickly as possible. Therefore, the input port /STB signal is connected to the /INT0 pin (as well as U4's clock pin and /IDS). Interrupt 0 is programmed in the interrupt priority register to be at the lower interrupt level so it cannot prevent refresh service. The interrupt 0 service routine stores the input character at the next location in the DRAM array, using the technique of a circular FIFO buffer. The routine also sends back an acknowledge pulse by clearing and setting the P3.4 pin, and then clears the BUSY (BFLAG) pin by performing a dummy read of port 6 (unless this character caused the buffer to be completely full).

During periods of access to the DRAM array by the input and output routines, the global interrupt enable bit (EA) is cleared so that the refresh interrupt does not disturb the contents of ports 1 and 4, or the /RAS, /CAS, and /WR signals.

The printer (output port) service routine runs all the time, except when the CPU is called to service the other conditions, therefore having the lowest priority. If there are characters in the buffer, polling is used to check for output port BUSY status. If the printer is not busy, then the character is sent, and the output port /STB pin (P4.3) is cleared and set. The output port /ACK line is connected to the /INT1 pin, so that the negative going edge of the /ACK signal is recorded as an interrupt pending. A very short INT1 service routine sets a software flag to indicate that the printer acknowledge the last character.

Possible Enhancements

There are a number of features that could be added to this design. As mentioned previously, the microcontroller could be put into the idle mode when the buffer is empty, conserving power.

The software could be enhanced to provide features such as multiple copies of a document, data compression, data conversion, automatic printer setup, etc. The PC operating system could be suitably modified to send a header for each file to be printed, containing these parameters. There is plenty of room for operating firmware expansion, and plenty of horsepower left in the 87C451 to handle these features.

The two serial port pins RxD and TxD were deliberately left unused so that input and/or output ports are easily implemented for serial interfaces or printers using the built-in UART. The pins used for parallel port handshaking could then be used as serial handshaking lines, providing the standard "modem" signals.

Combining the above two features, this circuit could act as a "splitter." By connecting a daisy-wheel printer to the serial port, a dot-matrix printer to the parallel port, and sending an "address" flag in the file header, simultaneous letter-quality and draft printing could be done.

The size of the DRAM array is easily expanded to one megabyte or large devices by connecting the additional address pins to port 4 bits 1 and 2. Only slight modifications to the operating firmware would be required.

Conclusion

The SC8XC451 microcontrollers provide plenty of I/O pins that previously had to be implemented by clumsy I/O expansion methods. The flexibility of port 6 means that this device can be used in a wide variety of applications requiring special port functions, while still using the industry standard 8051 instruction set.

The Application Note, describing a typical parallel printer buffer, makes full use of the 8XC451 features, yet allows room for enhancement and expansion.

256k Centronics printer buffer using the 87C451 microcontroller

AN417

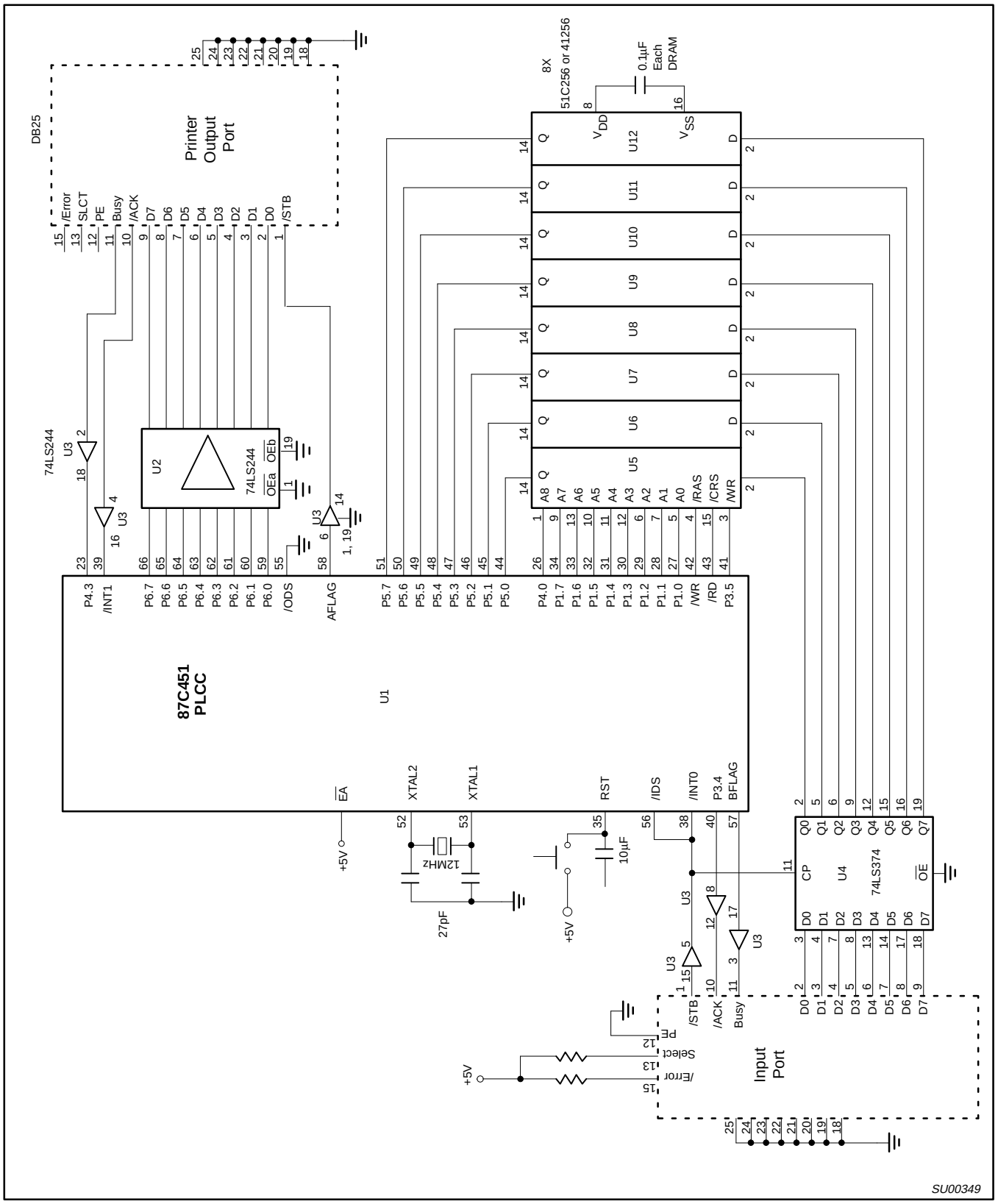

Figure 1. Schematic Diagram

256k Centronics printer buffer using the 87C451 microcontroller

AN417

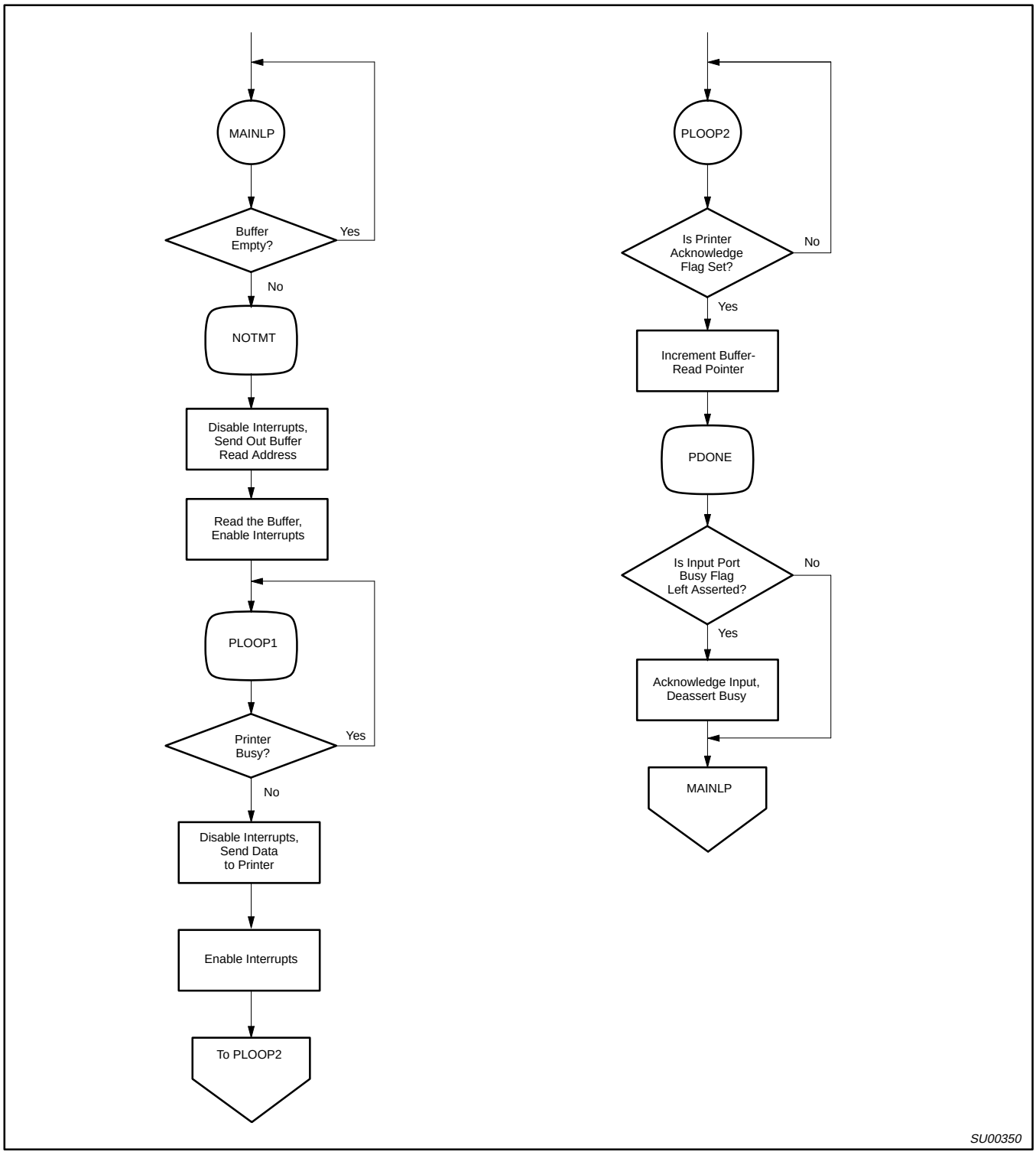

Figure 2. Flowchart of Transmit Operation

256k Centronics printer buffer using the 87C451 microcontroller

AN417

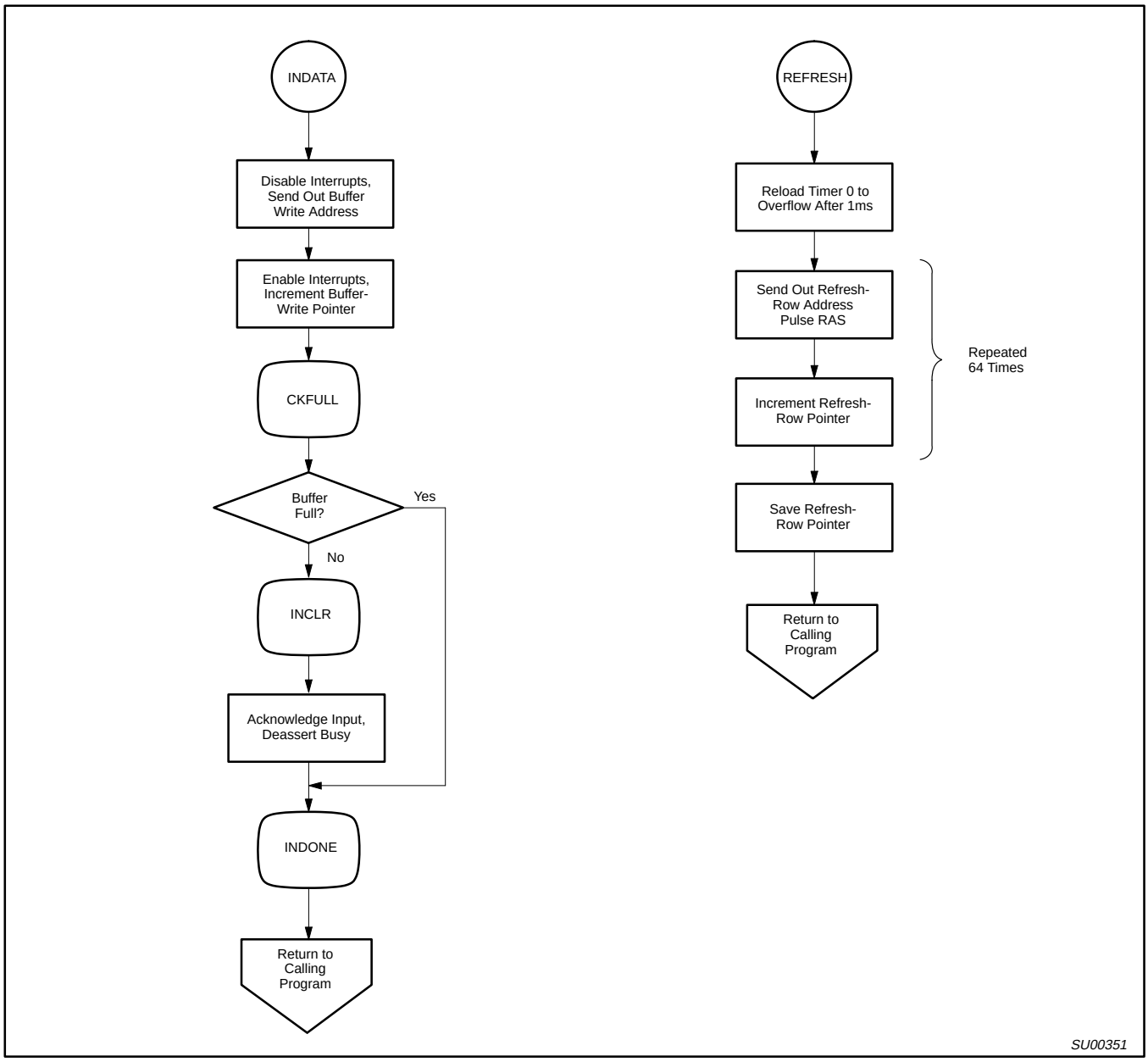

Figure 3. Flowchart of Receive and Refresh Operation

```
    XMIT:       MOV DPTR,8001H
    TEST:       MOVX A,@DPTR        ;READ THE CSR
                JB ACC.0,TEST       ;TEST IBF FLAG
;
;*****************************************************************************************;
;                                                                                         ;
;   256K PRINTER BUFFER PROGRAM USING THE 8xC451                                          ;
;      FOR CENTRONICS PARALLEL PRINTER PORTS                                              ;
;                                                                                         ;
;                       PHILIPS SEMICONDUCTORS                                            ;
;                            OCTOBER, 1988                                                ;
;                                                                                         ;
;*****************************************************************************************;
;
$Mod451
$Title(*XC451 Printer Buffer)
$Date(10/28/88)
;
;   PORT USAGE:
;
;   P0              Not used (reserved for data/address bus when external
;                     program memory is used).
;   P1              Lower 8 bits of DRAM address (A0 – A7).
;   P2              Not used (reserved for high-order address bus when external
;                     program memory is used).
;
;   P3.0            (Reserved for serial port.)
;   P3.1            (Reserved for serial port.)
;   P3.2 (/INT0)    Input port strobe input (interrupt).
;   P3.3 (/INT1)    Output port acknowledge input (interrupt).
;   P3.4            Input port acknowledge output.
;   P3.5            DRAM write enable output.
;   P3.6 (/WR)      DRAM row address select output.
;   P3.7 (/RD)      DRAM column address select output.
;
;   P4.0            Upper bit of DRAM address (A8).
;   P4.1            Reserved as an extra address line for 1 megabit DRAMS.
;   P4.2            Not used.
;   P4.3            Output port busy input (OBUSY).
;   P4.4–P4.7       Unused (not available on 64-pin DIP package).
;
;   P5              DRAM output data.
;
;   P6              Parallel output port.
;   /IDS            Input port strobe input (ISTB).
;   BFLAG           Input port busy output (IBUSY).
;
;   AFLAG           Output port strobe output (OSTB).
;   /ODS            Port 6 output enable, tied low.
;
;
;   Internal Register/RAM Usage:
;
REFCNT      EQU     020h            : Low order refresh byte.;
;
;   The following refer to the circular FIFO buffer
;   implemented in the DRAM array.
;
INLOW       EQU     22h             ; Incoming address low byte.
INMID       EQU     23h             ; Incoming address mid byte.
INHI        EQU     24h             ; Incoming address high byte.
OUTLOW      EQU     25h             ; Outgoing address low byte.
OUTMID      EQU     26h             ; Outgoing address mid byte.
OUTHI       EQU     27h             ; Outgoing address high byte.
;
OACK        EQU     28h             ; Holds flag for output port acknowledge.
FOACK       BIT     OACK.0          ; Bit-address of output port acknowledge flag.
;
```

```
;    Miscellaneous Equates:
;
TIME        EQU     –1000           ; Value for 1000 timer clocks = 1 millisecond.
TIMEHI      EQU     HIGH TIME       ; High byte of timer value.
TIMELO      EQU     LOW TIME        ; Low byte of timer value.
RAS         BIT     P3.6            ; DRAM column address select.
CAS         BIT     P3.7            ; DRAM row address select.
DRAMWR      BIT     P3.5            ; DRAM write control line.
IACK        BIT     P3.4            ; Input port ACK output.
ISTB        BIT     P3.2            ; Input port strobe line (INT0).
OBUSY       BIT     P4.3            ; Output port BUSY input.
OSTB        BIT     MA0             ; Output port strobe (MA0 bit in port 6 CSR).
;
;*****************************************************************************************************************
;
;    Reset and Interrupt Jump Table
;
                ORG     00h             ; Power-on reset.
                AJMP    START
;
                ORG     03h             ; INT 0.
                AJMP    INDATA          ; Data at input port.
;
                ORG     0Bh             ; Timer 0.
                AJMP    REFRESH         ; Refresh DRAM array.
;
                ORG     13h             ; INT 1.
                AJMP    OPACK           ; Output port acknowledge.
;*****************************************************************************************************************
;
;    Power up reset routine:
;    Set up refresh timer, enable timer interrupt and
;    external interrupt, initialize circular buffer pointers.
;
                ORG     18h
START:          MOV     SP,#40h         ; Initialize stack pointer.
                MOV     A,#00
;               MOV     REFCNT,A        ; Initialize refresh counter.
                MOV     INLOW,A         ; Initialize FIFO pointers.
                MOV     INMID,A
                MOV     INHI,A
                MOV     OUTLOW,A
                MOV     OUTMID,A
                MOV     OUTHI,A
;
;Initialize interrupt priority register so that DRAM refresh
;    (TF0) gets high priority, input port service (IE0) and output
;    port acknowledge service get lower priority. All other
;    interrupts set to lower priority level.
;
                MOV     IE,#00000111b   ; Timer0, INT0, and INT1 enabled.
                MOV     IP,#00000010b   ; Timer0 high priority.
                MOV     TL0,#TIMELO
                MOV     TH0,#TIMEHI
                MOV     TMOD,#00000001b ; Operate Timer0 in mode 1.
                MOV     TCON,#00010101b ; Timer0 run, I0 and I1 = edge.
;
;
```

```
;   Initialize Port 6 Control and Status Register.
;   –       'BFLAG' mode set to output value of IBF
;   –        (input port BUSY signal : IBUSY)
;   –       'AFLAG' set as logic 1 output
;            (output port strobe signal : OSTB)
;   –       'IDS' active on negative level
;            (input port strobe signal : ISTB)
;           MOV     CSR,#10011100b
            MOV     A,P6            ; Dummy read of P6 to clear IBF (IBUSY).
            SETB    EA              ; Enable interrupts.
;
;*****************************************************************************************************************
;
;   Main Routine:
;   Executes while not performing DRAM refresh or servicing
;   input port interrupt.
;
;   Check if buffer is not empty by comparing input and output
;   pointers. If not empty, go to NOTMT to output a byte.
;
MAINLP:     MOV     A,INLOW         ; Compare pointers.
            CJNE    A,OUTLOW,NOTMT
            MOV     A,INMID
            CJNE    A,OUTMID,NOTMT
            MOV     A,INHI
            CJNE    A,OUTHI,NOTMT
            SJMP    MAINLP
;
;   Buffer is not empty: compute row & column addresses for
;   a read cycle from DRAM.
;
NOTMT:      MOV     R4,OUTLOW       ; Save low byte of row.
            MOV     R5,OUTMID       ; Save upper bit of row.
            MOV     A,OUTHI         ; Shift to align correctly.
            RRC     A
            MOV     R7,A            ; Save upper column bit.
            MOV     A,OUTMID        ; Get low byte of column.
            RRC     A               ; Shift in bit from OUTHI.
            MOV     R6,A            ; Save.
;
;   Now do actual DRAM access to get the data byte at computed
;   address. Disable interrupts so we don't lose what we put
;   out on the ports.
;
            CLR     EA              ; Disable interrupts.
            MOV     P1,R4           ; Low byte row address.
            MOV     A,R5            ; Get high byte row address.
            ORL     A,#0FEh         ; Make sure OBUSY stays high.
            MOV     P4,A
            CLR     RAS             ; /RAS low.
            MOV     P1,R6           ; Low byte column address.
            MOV     A,R7            ; High byte column address.
            ORL     A,#0FEh         ; Make sure OBUSY stays high.
            MOV     P4,A
            CLR     CAS             ; /CAS low.
            MOV     R4,P5           ; Get the data byte
;
            SETB    CAS             ; /CAS high.
            SETB    RAS             ; /RAS high.
            CLR     FOACK           ; Clear acknowledge flag.
            SETB    EA              ; Re-enable interrupts.
;
```

```
PLOOP1:     JB      OBUSY,PLOOP1    ; Loop if printer busy.
;
            CLR     EA              ; Disable interrupts.
            MOV     P6,R4           ; Move byte to output port.
            CLR     MA0             ; Assert output port strobe.
            NOP                     ; Kill some time.
            NOP
            NOP
            NOP
            SETB    MA0             ; De-assert output port strobe.
            SETB    EA              ; Re-enable interrupts.
;
;   Following waits for /ACk to occur on output port. Loops on
;   acknowledge flag which is set by INT1 service routine when
;   /ACK occurs.
;
PLOOP2:     JNB     FOACK,PLOOP2    ; Wait till /ACK occurs.
;
            INC     OUTLOW          ; Increment output buffer pointer.
            MOV     A,OUTLOW
            CJNE    A,#00,PDONE
            INC     OUTMID
            MOV     A,OUTMID
            CJNE    A,#00,PDONE
            MOV     A,OUTHI
            INC     A
            ANL     A,#03h          ; Eliminate unused address bits
            MOV     OUTHI,A         ; and save.
;
;   Check if input port busy flag was left asserted, indicating that
;   the buffer was full after last input. If so, acknowledge input
;   port and de-assert input busy signal.
;
PDONE:      JNB     IBF,MAINLP      ; Not busy, return to main loop.
            CLR     EA              ; Disable interrupts.
            CLR     IACK            ; Assert /IACK.
            NOP                     ; Wait 7 microseconds.
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            MOV     A,P6            ; Dummy read of P6 clears IBF (IBUSY).
            NOP                     ; Wait 5 microseconds.
            NOP
            NOP
            NOP
            NOP
            SETB    IACK            ; De-assert /IACK.
            SETB    EA              ; Re-enable interrupts.
            AJMP    MAINLP          ; Return to main loop.
;
```

```
;*************************************************************************************************************
;
;    Interrupt 1 Service Routine:
;
;    –    Called when output port asserts /ACK.
;    –    Sets FOACK flag and returns.
;
OPACK:                  SETB            FOACK
        RETI
;
;*************************************************************************************************************
;
;    DRAM Refresh (Timer0) Interrupt Service:
;
;    –    Called once every millisecond by timer interrupt.
;    –    Refreshes 64 rows and then returns.
;    –    Therefore refreshes all rows every 4 milliseconds.
;         (Note that 41256/51C256 DRAM only requires a 256 row refresh.)
;
REFRESH:    PUSH    PSW
            MOV     TH0,#TIMEHI     ; Reload timer registers.
            MOV     TL0,#TIMELO
;
            MOV     P1,REFCNT       ; Get next row to refresh.
            MOVX    @R0,A           ; Pulse /RAS (/WR).
            INC     P1
            MOVX    @R0,A           ; 1
            INC     P1
            MOVX    @R0,A           ; 2
            INC     P1
            MOVX    @R0,A           ; 3
            INC     P1
            MOVX    @R0,A           ; 4
            INC     P1
            MOVX    @R0,A           ; 5
            INC     P1
            MOVX    @R0,A           ; 6
            INC     P1
            MOVX    @R0,A           ; 7
            INC     P1
            MOVX    @R0,A           ; 8
            INC     P1
            MOVX    @R0,A           ; 9
            INC     P1
            MOVX    @R0,A           ; 10
            INC     P1
            MOVX    @R0,A           ; 11
            INC     P1
            MOVX    @R0,A           ; 12
            INC     P1
            MOVX    @R0,A           ; 13
            INC     P1
            MOVX    @R0,A           ; 14
            INC     P1
            MOVX    @R0,A           ; 15
            INC     P1
            MOVX    @R0,A           ; 16
            INC     P1
            MOVX    @R0,A           ; 17
            INC     P1
            MOVX    @R0,A           ; 18
            INC     P1
            MOVX    @R0,A           ; 19
            INC     P1
            MOVX    @R0,A           ; 20
            INC     P1
```

```
MOVX    @R0,A           ; 21
INC     P1
MOVX    @R0,A           ; 22
INC     P1
MOVX    @R0,A           ; 23
INC     P1
MOVX    @R0,A           ; 24
INC     P1
MOVX    @R0,A           ; 25
INC     P1
MOVX    @R0,A           ; 26
INC     P1
MOVX    @R0,A           ; 27
INC     P1
MOVX    @R0,A           ; 28
INC     P1
MOVX    @R0,A           ; 29
INC     P1
MOVX    @R0,A           ; 30
INC     P1
MOVX    @R0,A           ; 31
INC     P1
MOVX    @R0,A           ; 32
INC     P1
MOVX    @R0,A           ; 33
INC     P1
MOVX    @R0,A           ; 34
INC     P1
MOVX    @R0,A           ; 35
INC     P1
MOVX    @R0,A           ; 36
INC     P1
MOVX    @R0,A           ; 37
INC     P1
MOVX    @R0,A           ; 38
INC     P1
MOVX    @R0,A           ; 39
INC     P1
MOVX    @R0,A           ; 40
INC     P1
MOVX    @R0,A           ; 41
INC     P1
MOVX    @R0,A           ; 42
INC     P1
MOVX    @R0,A           ; 43
INC     P1
MOVX    @R0,A           ; 44
INC     P1
MOVX    @R0,A           ; 45
INC     P1
MOVX    @R0,A           ; 46
INC     P1
MOVX    @R0,A           ; 47
INC     P1
MOVX    @R0,A           ; 48
INC     P1
MOVX    @R0,A           ; 49
INC     P1
MOVX    @R0,A           ; 50
INC     P1
MOVX    @R0,A           ; 51
INC     P1
MOVX    @R0,A           ; 52
INC     P1
MOVX    @R0,A           ; 53
INC     P1
MOVX    @R0,A           ; 54
INC     P1
```

```
                MOVX    @R0,A           ; 55
                INC     P1
                MOVX    @R0,A           ; 56
                INC     P1
                MOVX    @R0,A           ; 57
                INC     P1
                MOVX    @R0,A           ; 58
                INC     P1
                MOVX    @R0,A           ; 59
                INC     P1
                MOVX    @R0,A           ; 60
                INC     P1
                MOVX    @R0,A           ; 61
                INC     P1
                MOVX    @R0,A           ; 62
                INC     P1
                MOVX    @R0,A           ; 63
                INC     P1
;
                INC     P1              ; Adjust for next time
                MOV     REFCNT,P1       ; and save.
                POP     PSW
                RETI
;
;*****************************************************************************************************************
;
;   Data at Input Port:
;
;   This routine is called via interrupt INT0 whenever data
;   is strobed into the input port. It saves the data into the
;   DRAM array and increments the input pointer. If the output
;   pointer is now equal to the input pointer, then the buffer
;   is full, and we leave the busy flag set so that no more
;   data can be input until some is output and the buffer is
;   no longer full.
;
;
INDATA:     PUSH    PSW
                PUSH    ACC
                MOV     R1,INLOW        ; Lower 8 bits of row to R1.
                MOV     R2,INMID        ; Upper bit of row to R2.
                MOV     A,INHI          ; Get upper 2 bits.
                RRC     A               ; LSB to carry.
                MOV     R0,A
                MOV     A,INMID
                RRC     A               ; Shift bit into MSB.
                MOV     R3,A            ; Save.
;
                CLR     EA              ; Disable interrupts.
                MOV     P1,R1           ; LSB Row address.
                MOV     A,R2            ; MSB row address.
                ORL     A,#0FEh         ; Make sure OBUSY stays high.
                MOV     P4,A            ; MSB row address.
STBLP:      JNB     ISTB,STBLP      ; Check for end of strobe before DRAM write.
                CLR     RAS             ; /RAS low.
                CLR     DRAMWR          ; /WR low.
                MOV     P1,R3           ; LSB column address.
                MOV     1,R0            ; MSB column address.
                ORL     A,#0FEh         ; Make sure OBUSY stays high.
                MOV     P4,A            ; MSB column address.
                MOVX    A,@R0           ; Pulse /CAS low.
                SETB    RAS             ; /RAS high.
                SETB    DRAMWR          ; /WR high.
                SETB    EA              ; Re-enable interrupts.
;
```

```
                INC     INLOW           ; Increment input buffer pointer.
                MOV     A,INLOW
                CJNE    A,#00,CKFULL
                INC     INMID
                MOV     A,INMID
                CJNE    A,#00,CKFULL
                MOV     A,INHI
                INC     A
                ANL     A,#03h          ; Eliminate unused address bits.
                MOV     INHI,A
;
;   Compare input pointer to output pointer to see if the buffer is full.
;
CKFULL:         MOV     A,INLOW
                CJNE    A,OUTLOW,INCLR
                MOV     A,INMID
                CJNE    A,OUTMID,INCLR
                MOV     A,INHI
                CJNE    A,OUTHI,INCLR
;
;   If we get here, the buffer is full, so skip the acknowledge pulse.
;
                SJMP    INDONE
;
;
;   Send acknowledge pulse on /IACK line for 7 microseconds,
;   de-assert input BUSY signal halfway through.
;
INCLR:          CLR     EA              ; Disable interrupts.
                CLR     IACK            ; Assert /IACK.
                NOP                     ; Wait 7 microseconds.
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                MOV     A,P6            ; Dummy read of P6 clears IBF (IBUSY).
                NOP                     ; Wait 5 microseconds before clearing /IACK.
INDONE:         POP     ACC
                POP     PSW
                SETB    IACK            ; De-assert /IACK.
                SETB    EA              ; Re-enable interrupts.
                RETI
;
                END
```