Signetics 2650 Microprocessor application memos currently available:

AS50	Serial Input/Output
AS51	Bit and Byte Testing Procedures
AS52	General Delay Routines
AS53	Binary Arithmetic Routines
AS54	Conversion Routines
SP50	2650 Evaluation Printed Circuit Board Level System (PC1001)
SP51	2650 Demo Systems
SP52	Support Software for use with NCSS Timesharing System
SP53	Simulator, Version 1.2
SP54	Support Software for use with the General Electric Mark III Timesharing System
SS50	PIPBUG
SS51	Absolute Object Format (Revision 1)
MP51	2650 Initialization
MP52	Low Cost Clock Generator Circuits
MP53	Address and Data Bus Interfacing Techniques

signetics

INTRODUCTION

Interfacing a microprocessor to peripheral devices is an important part of a total microcomputer system design. The characteristics of the Interface depend to a large extent on total system requirements and other factors such as CPU loading and data speed. The use of interrupts and/or DMA structures also have an impact on the system input/output structure. The design of an I/O interface is not limited to hardware, and hardware/software trade-offs must be considered.

This applications memo examines the use of the 2650's versatile set of I/O instructions and the interface between the 2650 and I/O ports. Interrupt and DMA-controlled I/O are not discussed. A number of application examples for both serial and parallel I/O are given. Several types of input, output, and bidirectional interface devices are also examined.

Basic I/O Structure of the 2650

The 2650 is equipped with extensive and versatile input and output facilities. It can perform both single bit input/output and 8-bit parallel input/output.

The single bit input and output, called Sense (pin 1) and Flag (pin 40), are associated with the Program Status Word Upper (PSWU). The Flag output always reflects the value of bit 6 of the PSWU, while bit 7 of the PSWU always reflects the value of the Sense input signal. The Sense and Flag signals can be monitored and controlled with the PSW instructions.

Parallel I/O can be accomplished using the extended or non-extended read and write instructions. The extended and non-extended types are distinguished by the state of the E/$\overline{NE}$ output of the 2650.

The non-extended I/O instructions are single-byte instructions which accomplish a 1-byte data transfer into or out of the 2650. They also control the state of the D/$\overline{C}$ output, which can be used as a 1-bit device address in small systems.

The extended I/O instructions are 2-byte instructions. When executing extended I/O instructions, the second byte of the instruction is output on the lower 8 bits of the address bus (ADR0-ADR7). This information is normally used as an I/O device address to select 1 of up to 256 input or output devices, but may also be used to output control or status signals.

2650 I/O FACILITIES—GENERAL BLOCK DIAGRAM

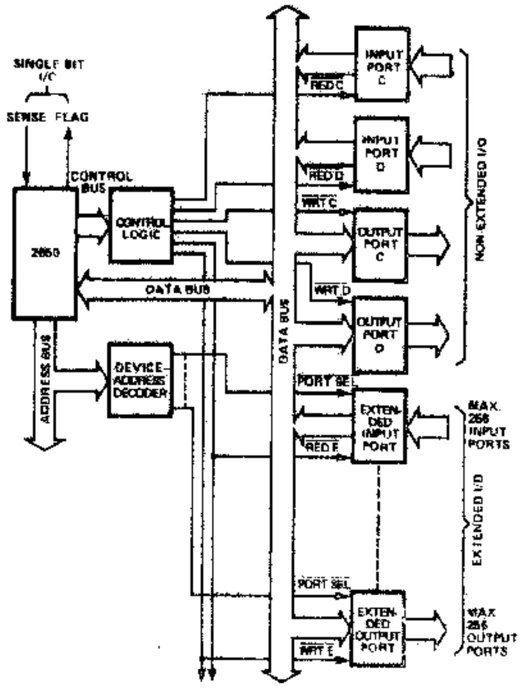


Figure 1

Parallel I/O operations may use any CPU register as the data source or destination. This offers significant flexibility in writing I/O software, because there is not a single accumulator register to create a "bottleneck" in the data flow. The functional block diagram in Figure 1 illustrates the various I/O facilities.

I/O As Part of the Memory Address Space

The 2650 user may choose to transfer data into or out of the processor using the memory control signals. The advantage of this technique is that the data can be read or written by the program with memory load and store instructions, and data may be directly operated upon with logical and arithmetic instructions. The memory referencing instructions can take advantage of the flexible addressing modes provided by the 2650, such as indexing and indirect addressing. A possible disadvantage of this method is that it may be necessary to decode more address lines to determine the device address than with the other I/O facilities.

To make use of this technique, the designer must assign memory addresses to I/O devices and design the device interfaces to respond to the same signals as memory.

I/O Interface Signals

Table I summarizes the state of the 2650 I/O interface signals for the various methods of I/O which are available.

SERIAL I/O USING THE SENSE INPUT AND FLAG OUTPUT

One of the I/O capabilities of the 2650 is provided by the sense input and flag output. The sense and flag pins may be used for single-bit input or output of status or control information. They can also be used to implement a serial data communications channel. Two examples of this application are given below.

Asynchronous Serial Communications Port

In applications where a serial type of terminal (like a teletypewriter) must be connected to the microcomputer system, the sense pin and flag pin can be used to interface with the terminal. The basic character format for asynchronous serial I/O is shown in Figure 2.

A number of parameters of this character format and the transmission speed, are different for various types of terminals. The variable parameters are:

Baud rate (bits per second): 110, 150, 300, 600, 1200, 2400, 4800, and 9600 baud.

Number of bits per character: 5, 6, 7, or 8 bits

Parity mode: even, odd, and no parity

Number of stop bits: 1 or 2

The control of the sense and flag pins for asynchronous serial I/O, with the appropriate parameters and baud rate, can be done completely with software. The hardware involved is limited to a simple line driver and receiver circuit which may be either an RS-232 interface or a 20mA current loop interface. The interface hardware is shown in Figure 3.

The software necessary to accomplish the serial I/O for a full-duplex line can be divided into 3 parts:

- The start bit detection and verification. After each start bit detection the start-bit level is verified for a low level at time intervals of 1/8 of 1 bit time. This prevents false start-bit recognition caused by line noise.
- The sampling of the data bits at the mid-bit time, echoing the data bit to the flag output, and loading the data bit into a CPU register.
- The input echo and check of parity bit and stop bits.

A timing diagram showing the start bit sampling and the bit echo appears in Figure 4.

TYPE OF I/O OPERATION	OPREQ	M/$\overline{IO}$	$\overline{R}$/W	ADR0-ADR7	ADR13 (E/$\overline{NE}$)	ADR14 (D/$\overline{C}$)
Sense (Input)	X	X	X	X	X	X
Flag (Output)	X	X	X	X	X	X
Extended Read	H	L	L	Second Byte of	H	X
Extended Write	H	L	H	Instruction	H	X
Non-Extended Read C	H	L	L	X	L	L
Non-Extended Read D	H	L	L	X	L	H
Non-Extended Write C	H	L	H	X	L	L
Non-Extended Write D	H	L	H	X	L	H
Memory I/O Read	H	H	L	ADR0-ADR7	ADR13	ADR14
Memory I/O Write	H	H	H	ADR0-ADR7	ADR13	ADR14

X = Don't Care

Table 1 I/O INTERFACE SIGNAL STATE

BASIC CHARACTER FORMAT FOR ASYNCHRONOUS I/O

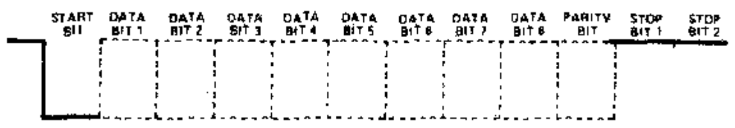


Figure 2

SERIAL I/O HARDWARE INTERFACES

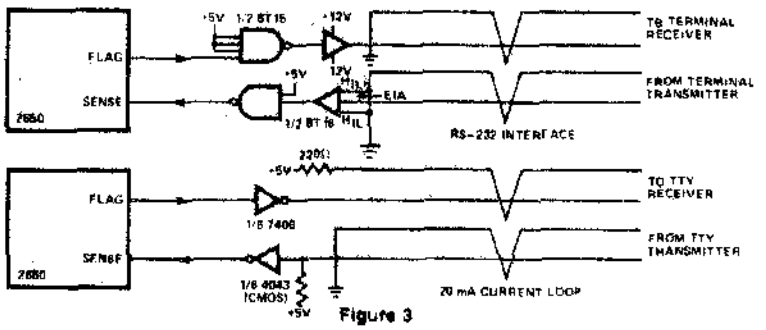


Figure 3

SERIAL I/O TIMING DIAGRAM

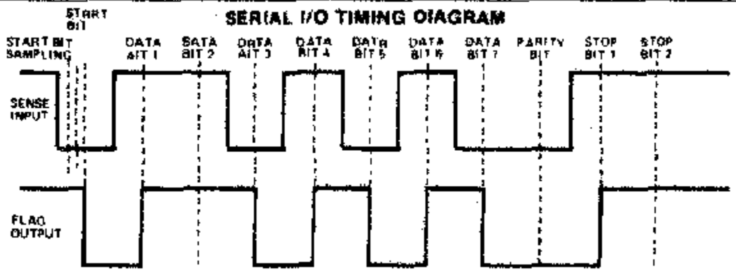


Figure 4

Three examples of the serial I/O routine with different speeds and parameters are presented in Figures 5 through 9. The bit and sample delay numbers (hexadecimal) in the definition listing (Figure 6) are for a CPU clock frequency of 1MHz. The hexadecimal delay numbers for a frequency of 1.25MHz are given in Table II. This table also lists the number of BDRR,R0 instructions that are necessary in the "bit delay and echo subroutine" to count cycles for the appropriate baud rate.

The examples of figures 7, 8, and 9 have the following parameters:

Figure 7: 110 baud, 7 data bits, avon parity and 1 stop bit.
Figure 9: 600 baud, 7 data bits, odd parity and 2 stop bits.
Figura 9: 2400 baud, 8 data bits, no parity and 1 stop bit.

The serial I/O routine uses 4 CPU registers (1 bank and R0) and affects 7 of the Program Status Word bits; namely, Sense, Flag, Overflow, Carry, Interdigit Carry, and the 2 Condition Code bits. The program also uses 1 level of the return address stack.

A parity error will set the Overflow bit, and a framing error (wrong stop bit level) will set the Interdigit Carry bit. At the end of the routine, the input character is stored in register R2

FLOWCHART OF THE SERIAL I/O ROUTINE

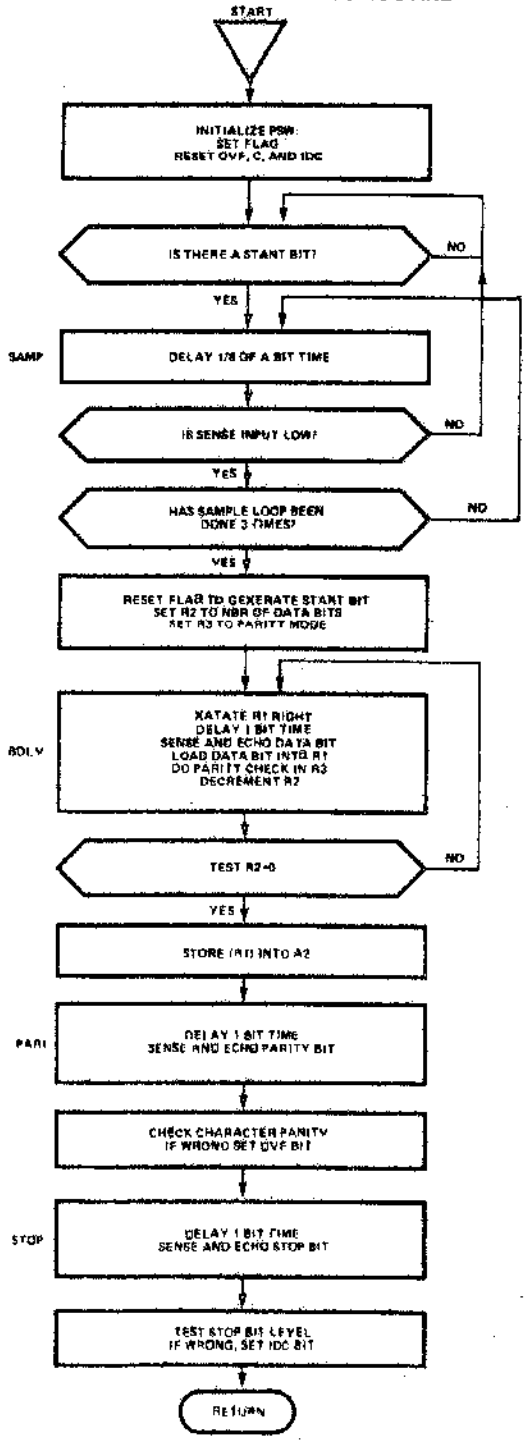

START
INITIALIZE PSW: SET FLAG RESET OVF, C, AND IDC
IS THERE A START BIT?
NO
YES
SAMP
DELAY 1/8 OF A BIT TIME
IS SENSE INPUT LOW?
NO
YES
HAS SAMPLE LOOP BEEN DONE 3 TIMES?
NO
YES
RESET FLAG TO GENERATE START BIT SET R2 TO NBR OF DATA BITS SET R3 TO PARITY MODE
BDLY
ROTATE R1 RIGHT DELAY 1 BIT TIME SENSE AND ECHO DATA BIT LOAD DATA BIT INTO R1 DO PARITY CHECK IN R3 DECREMENT R2
TEST R2=0
NO
YES
STORE (R1) INTO R2
PARI
DELAY 1 BIT TIME SENSE AND ECHO PARITY BIT
CHECK CHARACTER PARITY IF WRONG SET OVF BIT
STOP
DELAY 1 BIT TIME SENSE AND ECHO STOP BIT
TEST STOP BIT LEVEL IF WRONG, SET IDC BIT
RETURN

Figure 5

SERIAL I/O PARAMETER DEFINITIONS

```
TWIN ASSEMBLER VER 1 0                                   PAGE 0001

LINE ADDR  OBJECT  E SOURCE

0001                 * PV760051
0002                 ********************************************************
0003                 *                                                      *
0004                 *      **** PROGRAMMABLE SERIAL I/O ROUTINE ****       *
0005                 *                                                      *
0006                 * WITH THIS PROGRAM THE SENSE AND FLAG INPUT/OUTPUT OF *
0007                 * THE 2650 ARE USED TO INTERFACE WITH TERMINALS        *
0008                 * SUCH AS TTY, CRT TERMINALS, ETC. VIA THE BIT SERIAL  *
0009                 * ASYNCHRONOUS LINE DISCIPLINE                         *
0010                 *                                                      *
0011                 * ALL CHARACTER AND LINE PARAMETERS CAN BE MODIFIED    *
0012                 * SIMPLY IN THE SOFTWARE. THESE PARAMETERS ARE BAUD    *
0013                 * RATE, NUMBER OF DATA BITS, PARITY MODE AND STOP BITS *
0014                 *                                                      *
0015                 * THE PROGRAM HAS BEEN SET UP FOR A FULL DUPLEX LINE   *
0016                 * BUT CAN EASILY BE MODIFIED TO HALF DUPLEX MODE       *
0017                 *                                                      *
0018                 ********************************************************
0019                 *
0020                 * DEFINITIONS OF SYMBOLS
0021                 *
0022 0000            R0   EQU   0         PROCESSOR REGISTERS
0023 0001            R1   EQU   1
0024 0002            R2   EQU   2
0025 0003            R3   EQU   3
0026 0080            S    EQU   H'80'     PSU  SENSE
0027 0040            F    EQU   H'40'          FLAG
0028 0020            IDC  EQU   H'20'          INTERDIGIT CARRY
0029 0004            OVF  EQU   H'04'          OVERFLOW
0030 0001            C    EQU   H'01'          CARRY/BORROW
0031 0002            N    EQU   2              BRANCH CONDITION  NEGATIVE
0032 0003            UN   EQU   3                                UNCONDITIONAL
0033                 *
0034                 ************************************************************
0035                 *
0036                 * SOFTWARE DEFINITIONS OF BAUD RATE, CHARACTER FORMAT, PARITY,
0037                 * PARITY MODE, ETC
0038                 *
0039                 * NUMBER OF DATA BITS
0040                 *
0041 0008            D08  EQU   H'08'     CHARACTER HAS 8 DATA BITS
0042 0080            BP8  EQU   H'80'
0043 0007            D07  EQU   H'07'     CHARACTER HAS 7 DATA BITS
0044 0040            BP7  EQU   H'40'
0045 0006            D06  EQU   H'06'     CHARACTER HAS 6 DATA BITS
0046 0020            BP6  EQU   H'20'
0047 0005            D05  EQU   H'05'     CHARACTER HAS 5 DATA BITS
0048 0010            BP5  EQU   H'10'
0049                 *
0050                 * BIT DELAYS AT 1 MHZ CLOCK FREQUENCY
0051                 *
0052 00E6            B110 EQU   H'E6'     BIT DELAY AT 110 BAUD
0053 0069            B300 EQU   H'69'     BIT DELAY AT 300 BAUD
0054 0034            B600 EQU   H'34'     BIT DELAY AT 600 BAUD
0055 0017            B12  EQU   H'17'     BIT DELAY AT 1200 BAUD
0056 000A            B24  EQU   H'0A'     BIT DELAY AT 2400 BAUD
0057                 *
0058                 * START BIT  SAMPLE DELAYS AT 1 MHZ CLOCKFREQUENCY
0059                 *
0060 0045            S110 EQU   H'45'     SAMPLE DELAY AT 110 BAUD
0061 001A            S300 EQU   H'1A'     SAMPLE DELAY AT 300 BAUD
0062 000D            S600 EQU   H'0D'     SAMPLE DELAY AT 600 BAUD
0063 000B            S12  EQU   H'0B'     SAMPLE DELAY AT 1200 BAUD
0064 0005            S24  EQU   H'05'     SAMPLE DELAY AT 2400 BAUD
0065                 *
0066                 * PARITY MODE
0067                 *
0068 0000            EP   EQU   H'00'     EVEN PARITY
0069 00A0            OP   EQU   H'A0'     ODD PARITY
0070                 *
```

Figure 6

SERIAL I/O ASSEMBLY LISTING—EXAMPLE 1

```
TWIN ASSEMBLER VER 1 0                                   PAGE 0003

LINE ADDR  OBJECT  E SOURCE

0073                 ************************************************************
0074                 * EXAMPLE 5  FULL DUPLEX (BIT BY BIT ECHO), 110 BAUD,
0075                 * 7 DATA BITS, EVEN PARITY AND 1 STOP BIT
0076                 *
0077 0500                 ORG     H'0500'
0078 0500 7640       STRT PPSU    F          SET FLAG TO SWITCH OFF THE LINE
0079 0502 7525            CPSL    OVF+C+IDC
0080 0504 12         TEST SPSU               WAIT FOR START BIT
0081 0505 1A7D            BCTR,N  TEST
0082 0507 0603            LODI,R2 H'03'      SET R2 TO NUMBER OF SAMPLES
0083 0509 0545       SAMP LODI,R1 S110       SET R1 TO SAMPLE DELAY
0084 050B F97E            BDRR,R1 $
0085 050D 12              SPSU               TEST FOR START BIT VALIDITY
0086 050E 1A74            BCTR,N  TEST       IF NOT VALID, GO BACK TO TEST
0087 0510 FA77            BDRR,R2 SAMP
0088 0512 0700            LODI,R3 EP         SET R3 TO EVEN PARITY MODE
0089 0514 0607            LODI,R2 D07        SET R2 TO NUMBER OF DATA BITS
0090 0516 7440            CPSU    F          GENERATE START BIT
0091 0518 51         BITS RRR,R1
0092 0519 3B12            BSTR,UN BDLY       GO TO DELAY AND ECHO ROUTINE
0093 051B FA7B            BDRR,R2 BITS       TEST FOR NUMBER OF DATA BITS
0094 051D 01              LODZ    R1
0095 051E C2              STRZ    R2         LOAD R2 WITH CHARACTER
0096 051F 3B0C       PAR1 BSTR,UN BDLY
0097 0521 9A02            BCFR,N  STOP
0098 0523 7704            PPSL    OVF        IF WRONG PARITY, SET OVF
0099 0525 0700       STOP LODI,R3 0          CLEAR R3
0100 0527 3B04            BSTR,UN BDLY
0101 0529 16         EXI1 RETC,N             TEST STOP BIT LEVEL
0102 052A 7720            PPSL    IDC        IF WRONG, SET IDC BIT
0103 052C 17         EXI2 RETC,UN
0104                 *
0105                 ************************************************************
0106                 * BIT DELAY AND ECHO SUBROUTINE
0107                 *
0108 052D 05E6       BDLY LODI,R1 B110       LOAD R1 WITH BIT DELAY NUMBER
0109 052F F97E            BDRR,R1 $
0110 0531 F97E            BDRR,R1 $
0111 0533 F97E            BDRR,R1 $
0112 0535 F97E            BDRR,R1 $
0113 0537 12              SPSU               TEST INPUT LEVEL
0114 0538 1A04            BCTR,N  ONE
0115 053A 7440            CPSU    F          IF LOW, ECHO A ZERO
0116 053C 1B04            BCTR,UN BIT1
0117 053E 7640       ONE  PPSU    F          IF HIGH, ECHO A ONE
0118 0540 6540            IORI,R1 BP7        INSERT A ONE BIT INTO R1
0119 0542 21         BIT1 EORZ    R1
0120 0543 [illegible]     [illegible]        [illegible]
0121 0544 17              RETC,UN
0122                 *
0123 0000                 END

TOTAL ASSEMBLY ERRORS    0000
```

Figure 7

SERIAL I/O ASSEMBLY LISTING

EXAMPLE 2

```
TWIN ASSEMBLER VER 1.0                                    PAGE 0003

LINE ADDR OBJECT E SOURCE

                    ************************************************************
                    * EXAMPLE 2  FULL DUPLEX (BIT BY BIT ECHO)  600 BAUD
                    * 7 DATA BITS, ODD PARITY AND 2 STOP BITS
                    *
     0000                 ORG    H'0500'
     0500 7640      STRT PPSU   F          SET FLAG TO SWITCH OFF THE LINE
     0502 7528            CPSL   OVF+WC+IDC
     0504 12        TEST SPSU              WAIT FOR START BIT
     0505 1A7D            BCTR,N TEST
     0507 0603            LODI,R2 H'03'    SET R2 TO NUMBER OF SAMPLES
     0509 053C      SAMP LODI,R1 S006      SET R1 TO SAMPLE DELAY
     050B F97E            BDRR,R1 $
     050D 12              SPSU             TEST FOR START BIT VALIDITY
     050E 1A74            BCTR,N TEST      IF NOT VALID, GO BACK TO TEST
     0510 FA77            BDRR,R2 SAMP
     0512 0700            LODI,R3 0        SET R3 TO ODD PARITY MODE
     0514 0607            LODI,R2 007      SET R2 TO NUMBER OF DATA BITS
     0516 7440            CPSU   F         GENERATE START BIT
     0518 51        BITS RRR,R1
     0519 3B1A            BSTR,UN BDLY     GO TO DELAY AND ECHO ROUTINE
     051B FA7B            BDRR,R2 BITS     TEST FOR NUMBER OF DATA BITS
     051D 01              LODZ   R1
     051E C2              STRZ   R2        LOAD R2 WITH CHARACTER
     051F 3B14      PARI BSTR,UN BDLY
     0521 9802            BCFR,N STO1
     0523 7704            PPSL   OVF       IF WRONG PARITY, SET OVF
     0525 0700      STO1 LODI,R3 0         CLEAR R3
     0527 3B0C            BSTR,UN BDLY
     0529 1A02            BCTR,N STO2      TEST STOP BIT LEVEL
     052B 7720            PPSL   IDC       IF WRONG, SET IDC BIT
     052D 0700      STO2 LODI,R3 0         CLEAR R3
     052F 3B04            BSTR,UN BDLY
     0531 16        EXI1 RETC,N            TEST STOP BIT 2 LEVEL
     0532 7720            PPSL   IDC       IF WRONG, SET IDC BIT
     0534 17        EXI2 RETC,UN
                    *
                    ************************************************************
                    * BIT DELAY AND ECHO SUBROUTINE
                    *
     0535 0400      BDLY LODI,R0 BRKE      SET R0 TO BIT DELAY NUMBER
     0537 F87E            BDRR,R0 $
     0539 12              SPSU             TEST DATA BIT LEVEL
     053A 1A04            BCTR,N ONE
     053C 7440            CPSU   F         IF LOW, ECHO A ZERO
     053E 1B04            BCTR,UN BITL
     0540 7640      ONE  PPSU   F          IF HIGH, ECHO A ONE
     0542 6540            IORI,R1 BP       INSERT DATA BIT INTO R1
     0544 23        BITL EORZ   R3
     0545 C3              STRZ   R3        DO PARITY CHECK
     0546 17              RETC,UN
                    *
     0000                 END    0

TOTAL ASSEMBLY ERRORS = 0000
```

Figure 8

EXAMPLE 3

```
TWIN ASSEMBLER VER 1.0                                    PAGE 0002

LINE ADDR OBJECT E SOURCE

                    ************************************************************
                    * EXAMPLE 3  FULL DUPLEX (BIT BY BIT ECHO)  2400 BAUD
                    * 8 DATA BITS, NO PARITY AND 1 STOP BIT
                    *
     0000                 ORG    H'0500'
     0500 7640      STRT PPSU   F          SET FLAG TO SWITCH OFF THE LINE
     0502 7528            CPSL   OVF+WC+IDC
     0504 12        TEST SPSU              WAIT FOR START BIT
     0505 1A7D            BCTR,N TEST
     0507 0603            LODI,R2 H'03'    SET R2 TO NUMBER OF SAMPLES
     0509 0504      SAMP LODI,R1 D04       SET R1 TO SAMPLE DELAY
     050B F97E            BDRR,R1 $
     050D 12              SPSU             TEST FOR START BIT VALIDITY
     050E 1A74            BCTR,N TEST      IF NOT VALID, GO BACK TO TEST
     0510 FA77            BDRR,R2 SAMP
     0512 0608            LODI,R2 D08      SET R2 TO NUMBER OF DATA BITS
     0514 7440            CPSU   F         GENERATE START BIT
     0516 51        BITS RRR,R1
     0517 3B0C            BSTR,UN BDLY     GO TO DELAY AND ECHO ROUTINE
     0519 FA7B            BDRR,R2 BITS     TEST FOR NUMBER OF DATA BITS
     051B 01              LODZ   R1
     051C C2              STRZ   R2        LOAD R2 WITH CHARACTER
     051D 0700      STOP LODI,R3 0         CLEAR R3
     051F 3B04            BSTR,UN BDLY
     0521 16        EXI1 RETC,N            TEST STOP BIT LEVEL
     0522 7720            PPSL   IDC       IF WRONG, SET IDC BIT
     0524 17        EXI2 RETC,UN
                    *
                    ************************************************************
                    * BIT DELAY AND ECHO SUBROUTINE
                    *
     0525 0425      BDLY LODI,R0 BR24      SET R0 TO BIT DELAY NUMBER
     0527 F87E            BDRR,R0 $
     0529 12              SPSU             TEST DATA BIT LEVEL
     052A 1A04            BCTR,N ONE
     052C 7440            CPSU   F         IF LOW, ECHO A ZERO
     052E 1B04            BCTR,UN BITS
     0530 7640      ONE  PPSU   F          IF HIGH, ECHO A ONE
     0532 6580            IORI,R1 BP8      INSERT DATA BIT INTO R1
     0534 C3        BITS STRZ   R3
     0535 17              RETC,UN
                    *
     0000                 END    0

TOTAL ASSEMBLY ERRORS = 0000
```

Figure 9

Data String Output

A typical application for the flag output is a data string output. The advantage of this output method is that it can provide a large number of output bits with little address or control logic decoding. For example, this method can be used to output data for an array of numeric displays, single bit indicators, or column drivers of a parallel numeric printer. An example of the hardware required to implement this type of output channel is given in Figure 10.

Here, the Address 14 output is used as a data strobe signal. However, the data strobe signal could also be built up by decoding more address bits so that the system memory size would not be limited to 16K bytes as in this example.

A listing of the program required is given in Figure 14. The data is assumed to be located in the system's RAM as illustrated in Figure 11.

The least-significant bit of the least-significant byte will be output first. The table length (TLEN) and the number of bits per byte (BPW) can be adapted as necessary by software modifications. The data strobe pulse on output ADR14 is generated by doing the dummy instruction STRA,R0 to address H'4000'.

BAUD RATE	SAMPLE DELAY NUMBER AT 1.25MHz	BIT DELAY NUMBER AT 1.25MHz	NUMBER OF BDRR,R0 INSTRUCTIONS AT 1.25MHz	NUMBER OF BDRR,R0 INSTRUCTIONS AT 1MHz
110	D0	E5	5	4
300	4A	C5	2	2
600	24	DE	1	1
1200	11	6A	1	1
2400	07	30	1	1

Table 2 BIT DELAY PROGRAM CONSTANTS AT A CLOCK FREQUENCY OF 1.25MHz (HEXADECIMAL)

INTERFACE DIAGRAM FOR DATA STRING OUTPUT

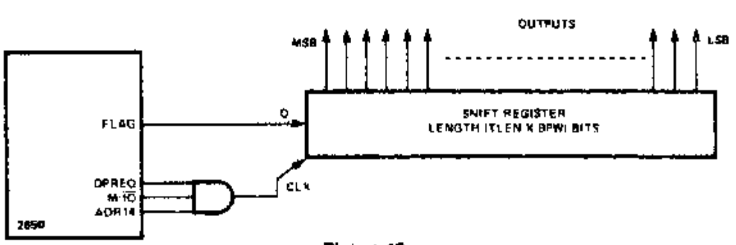


Figure 10

DATA ORGANIZATION FOR DATA STRING OUTPUT

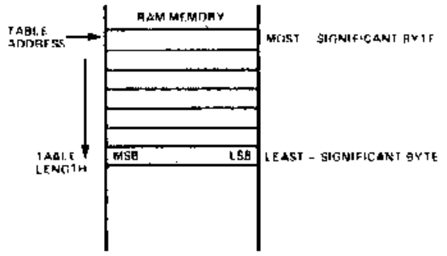


Figure 11

TIMING DIAGRAM OF DATA STRING OUTPUT ROUTINE

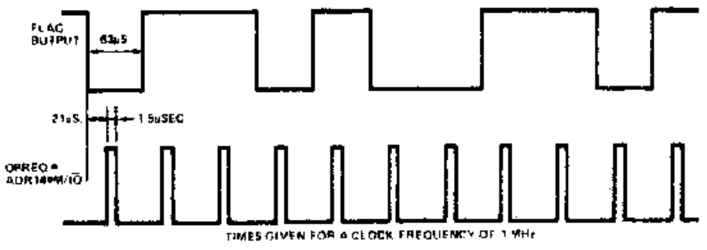


Figure 12

FLOWCHART OF DATA STRING OUTPUT ROUTINE

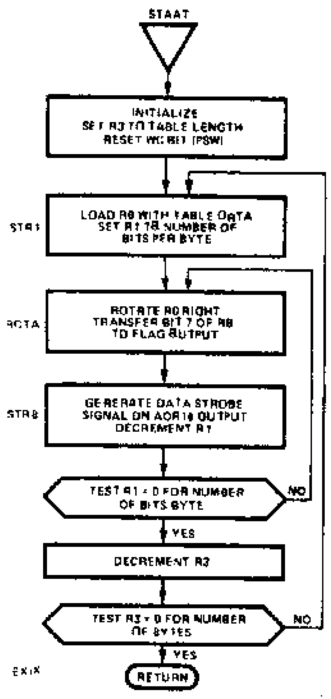
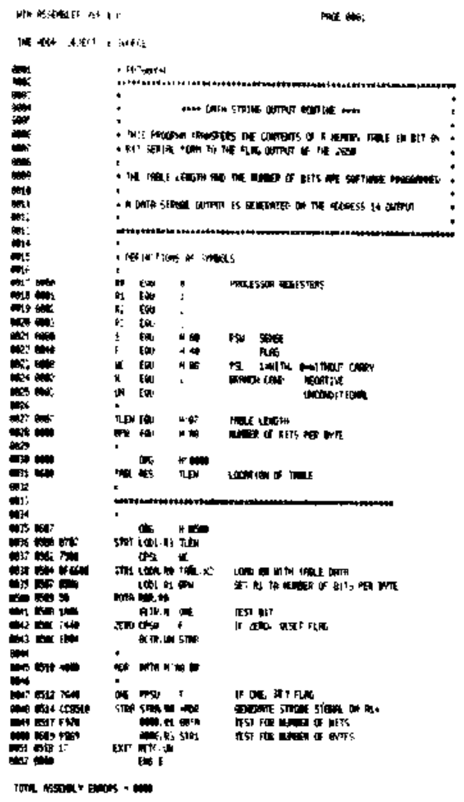

STAAT
INITIALIZE
SET R3 TO TABLE LENGTH
RESET WC BIT (PSW)
STR1
LOAD R0 WITH TABLE DATA
SET R1 TO NUMBER OF BITS PER BYTE
ROTA
ROTATE R0 RIGHT
TRANSFER BIT 7 OF R0 TO FLAG OUTPUT
STRB
GENERATE DATA STROBE SIGNAL ON ADR14 OUTPUT
DECREMENT R1
TEST R1 = 0 FOR NUMBER OF BITS BYTE
NO
YES
DECREMENT R3
TEST R3 = 0 FOR NUMBER OF BYTES
NO
YES
EXIT
RETURN

Figure 13

ASSEMBLY LISTING OF DATA STRING OUTPUT ROUTINE



```
PIP ASSEMBLER VER 1.0                                PAGE 0001

LINE ADDR OBJECT E SOURCE

                  * [illegible]
                  ***********************************************************
                  *                                                         *
                  *          **** DATA STRING OUTPUT ROUTINE ****           *
                  *                                                         *
                  * THIS PROGRAM TRANSFERS THE CONTENTS OF A MEMORY TABLE IN BIT BY *
                  * BIT SERIAL FORM TO THE FLAG OUTPUT OF THE 2650          *
                  *                                                         *
                  * THE TABLE LENGTH AND THE NUMBER OF BITS ARE SOFTWARE PROGRAMMED *
                  *                                                         *
                  * A DATA STROBE OUTPUT IS GENERATED ON THE ADDRESS 14 OUTPUT *
                  *                                                         *
                  ***********************************************************
                  *
                  * DEFINITIONS OF SYMBOLS
                  *
     0000         R0   EQU   0          PROCESSOR REGISTERS
     0001         R1   EQU   1
     0002         R2   EQU   2
     0003         R3   EQU   3
     0080         S    EQU   H'80'      PSU  SENSE
     0040         F    EQU   H'40'           FLAG
     000[illegible] WC EQU   H'0[illegible]' PSL  WITH/WITHOUT CARRY
     000[illegible] N  EQU   [illegible] BRANCH COND  NEGATIVE
     000[illegible] UN EQU   [illegible]              UNCONDITIONAL
                  *
     00[illegible] TLEN EQU  H'[illegible]' TABLE LENGTH
     00[illegible] BPB  EQU  H'[illegible]' NUMBER OF BITS PER BYTE
                  *
     0000              ORG   H'0000'
     0[illegible] TABL RES   TLEN       LOCATION OF TABLE
                  *
                  ***********************************************************
                  *
     0[illegible]      ORG   H'[illegible]'
     [illegible] [illegible] STRT LODI,R3 TLEN
     [illegible] [illegible]      CPSL  WC
     [illegible] [illegible] STR1 LODA,R0 TABL,R3  LOAD R0 WITH TABLE DATA
     [illegible] [illegible]      LODI,R1 BPB    SET R1 TO NUMBER OF BITS PER BYTE
     [illegible] [illegible] ROTA RRR,R0
     [illegible] [illegible]      BCTR,N ONE     TEST BIT
     [illegible] [illegible] ZERO CPSU  F        IF ZERO, RESET FLAG
     [illegible] [illegible]      BCTR,UN STRB
                  *
     [illegible] [illegible] NOP  DATA H'[illegible]'
                  *
     [illegible] [illegible] ONE  PPSU  F        IF ONE, SET FLAG
     [illegible] [illegible] STRB STRA,R0 +[illegible] GENERATE STROBE SIGNAL ON A14
     [illegible] [illegible]      BDRR,R1 ROTA   TEST FOR NUMBER OF BITS
     [illegible] [illegible]      BDRR,R3 STR1   TEST FOR NUMBER OF BYTES
     [illegible] [illegible] EXIT RETC,UN
     0000              END   0

TOTAL ASSEMBLY ERRORS = 0000
```

Figure 14

PARALLEL INPUT/OUTPUT

The 2650 instruction set contains the following six input/output instructions

		NO. BYTES
WRTC, RX	Write Control	1
REDC, RX	Read Control	1
WRTD, RX	Write Data	1
REDD, RX	Read Data	1
WRTE, RX DEVA	Write Extended	2
REDE, RX DEVA	Read Extended	2

The control signals generated by each I/O instruction simplify the interface circuitry required to generate I/O selection and timing signals. A low-cost control signal interface with related timing is shown in Figures 15 and 16.

When using standard TTL and 8T series I/O ports, the I/O operations can be done without slowing down the system. In this case the OPACK input could be controlled directly, for all I/O operations.

Non-Extended I/O

The single-byte I/O instructions of the 2650 are referred to as non-extended I/O. In small systems with only two 8-bit input ports and two 8-bit output ports, this I/O facility requires a minimum of hardware interfacing between the CPU and I/O ports. The signals $\overline{WRTC}$, $\overline{WRTD}$, $\overline{REDC}$, and $\overline{REDD}$ generated by the control logic decoder in Figure 15 can be used directly as output port clock pulses and input port enable signals, respectively.

Sequential I/O With Non-Extended I/O Instructions

In systems where a larger number of devices must be serviced in sequence, the use of a simple 8-bit output port can offer considerable savings in software. Normally the devices could be serviced with extended I/O instructions. However, since the device address is the second byte in this type of instruction, a series of data fetch and I/O instructions would be required to service the devices in sequence.

With an 8-bit output port functioning as a device address register, the device address can be modified under software control. In this way a simple program loop can serve up to 8 I/O ports by rotating a single '1' through a CPU register that is output as a device address. This I/O addressing technique may also be used advantageously in systems where I/O operation requests are detected by software polling. A functional block diagram of this technique is shown in Figure 17.

Extended I/O

There are 2 extended I/O instructions in the 2650 instruction set. In these 2-byte instructions the first byte specifies the operation code and the data source or destination register in the CPU. The second byte provides an 8-bit device address code that is output on the 8 least-significant bits of the address bus, ADR0 through ADR7.

The control signal decoding diagram (Figure 15) can be simplified for systems using only extended I/O, as shown in Figure 18. The timing diagram of Figure 16 also applies to this decoding technique.

Device Address Decoding Schemes

For extended I/O it is necessary to decode the address lines ADR0 through ADR7 in order to generate appropriate port selection signals. The choice of an address decoding scheme depends on factors such as total I/O requirements, the type of I/O ports used, and the total system configuration.

In principle, there are 2 basic methods of device address decoding. One method is the use of hardwired logic in which the device address is fixed; the other is a hardware programmable method in which the device addresses are individually set with jumpers or switches. Some examples of these methods are given in Figures 19 and 20.

In many applications a combination of these 2 methods is used. In addition the control logic can be implemented as an integral part of the device address decoding. An example is shown in Figure 21.

CONTROL SIGNAL INTERFACE USING THE 74(L) S138 DECODER

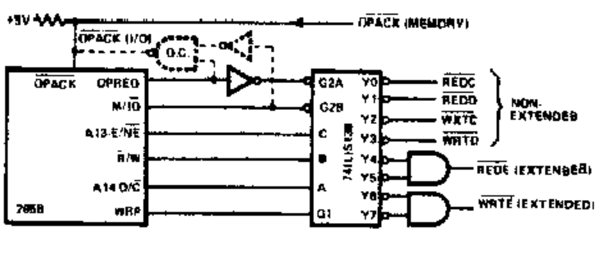


Figure 15

I/O INTERFACE TIMING DIAGRAMS (Figure 15)

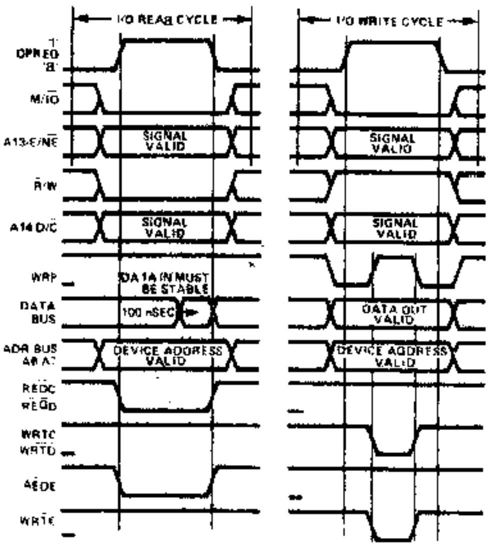


Figure 16

Memory Mapped I/O

In memory mapped I/O, the I/O devices are treated as memory locations. An advantage of this technique is that all memory referencing instruction types (store, load, arithmetic, logical, etc.) can be used directly for I/O data. Device address decoding is not necessarily more complex than for normal extended I/O, since all I/O addresses could be located in a specific address block. Of course, this technique can only be used in systems which do not use the full memory address space for programs. A diagram of the I/O control logic, using the ADR14 output to discriminate between memory and I/O operations, is given in Figure 22. The device address decoding methods described earlier can also be applied to memory mapped I/O.

SEQUENTIAL I/O TECHNIQUE

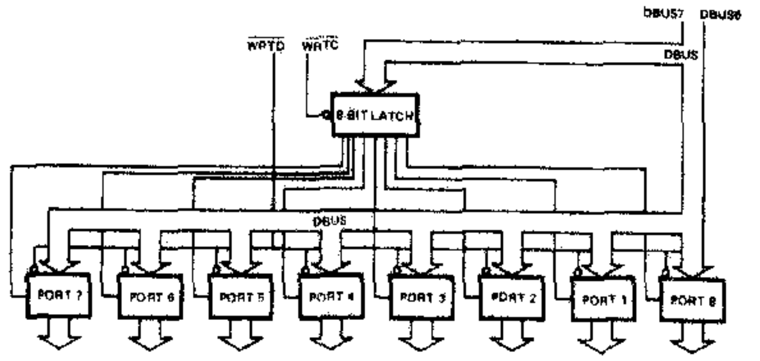
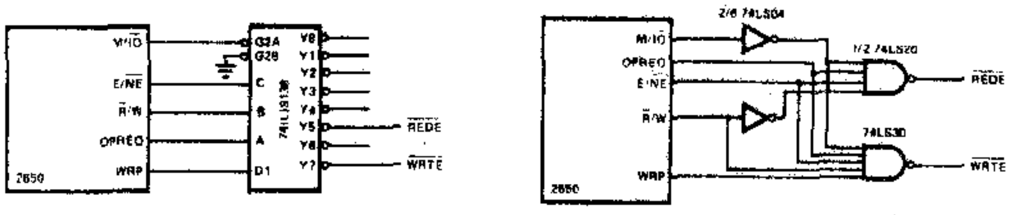


Figure 17

SIMPLIFIED CONTROL LOGIC WHEN USING EXTENDED I/O ONLY

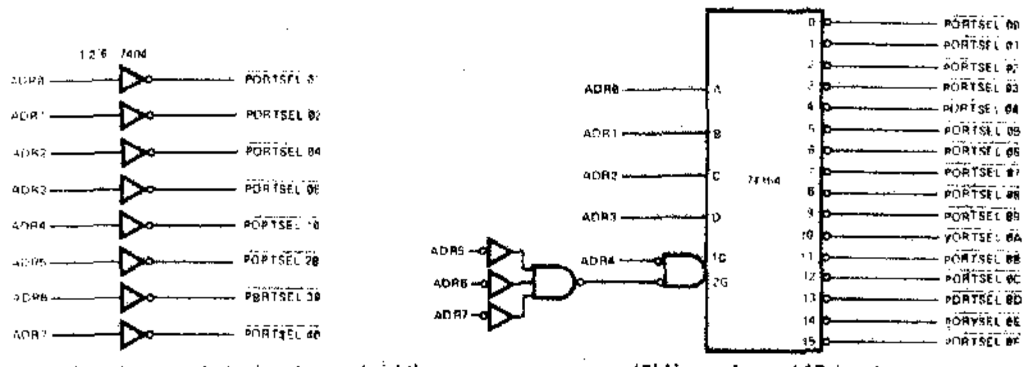


(A) Using 1-of-8 Decoder

(B) Using Logic Gates

Figure 18

SOME POSSIBLE TECHNIQUES FOR DEVICE ADDRESS DECODING

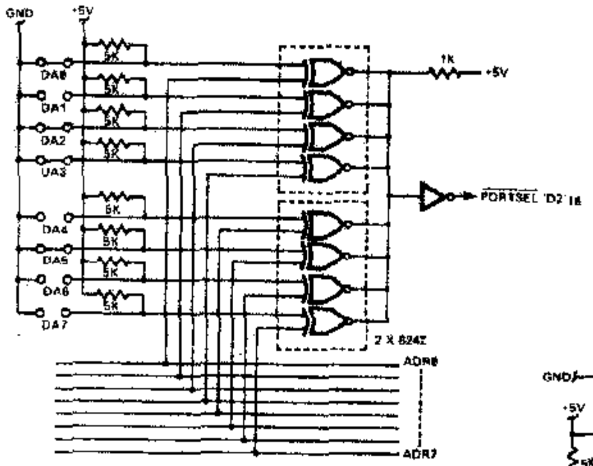


(A) Each address line selects one device (maximum of eight)

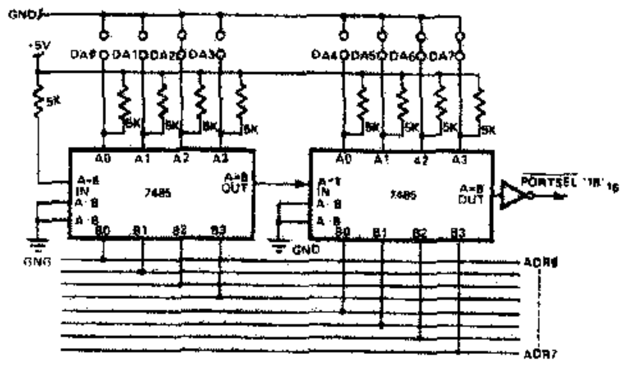
(B) Using a 1-out-of-16 decoder

Figure 19

HARDWARE PROGRAMMABLE DEVICE ADDRESS DECODERS



(A) Using Exclusive-OR Gates



(B) Using Comparators

Figure 20

COMBINED CONTROL LOGIC AND ADDRESS DECODING

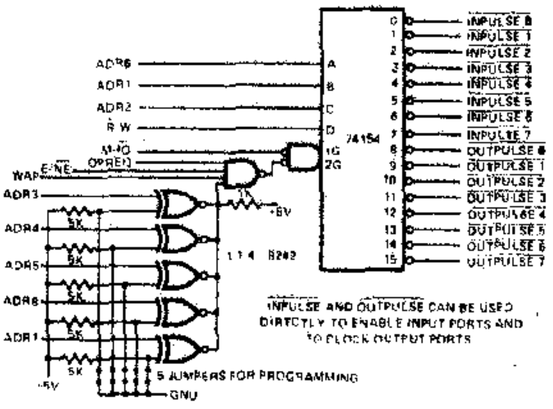


Figure 21

SINGLE POINT CONTROL

In many applications, the capability to set, clear, or test a single output point selected from a large number of output points is required. Designs of this type can be implemented using the 2650 I/O instructions. When used as described below, the WRTE, WRTC, and WRTD instructions become "set/clear single-bit" instructions, while the REDE instruction becomes a "test single-bit" instruction.

Single Bit Output—Direct Address

The write extended instruction can be used to select and set or clear a single output bit. The 2 bytes of the instruction can be interpreted as follows:

BYTE 0

1	1	0	1	0	1	X	X

BYTE 1

S/C	A_6	A_5	A_4	A_3	A_2	A_1	A_0

A through A_6 of the second byte specify the output selected. The S/C bit specifies whether the bit is set or cleared. A typical hardware configuration controlling 64 points is shown in Figure 23. Here, the control line decoding and partial address decoding is done by the 74LS138, which selects one of the eight 9334s. One of the 8 latches in the selected 9334 is enabled by ADR0, ADR1, and ADR2 and is either cleared or set, as determined by the value of ADR7.

The XX field in the first byte selects 1 of the 4 available registers and outputs its contents on the data bus. Since this information is not used in this application, the value of XX is not important. However, it could be used to output an 8-bit control or status word in conjunction with the set/clear operation.

Single Bit Output—Indirect Address

If the address of the output to be set or cleared must be determined at program run time, the WRTD and WRTC instructions can be used. The address of the output bit is first loaded into one of the 2650 registers. A WRTD, Rx instruction is then issued if the bit is to be set, and a WRTC, Rx instruction is issued if the bit is to be cleared. The bit select is output on the data bus, and the D/$\overline{C}$ output carries the set/clear information. The hardware implementation can be the same as shown in Figure 23 except that ADR0-ADR5 are replaced by DBUS0-DBUS5, and ADR7 is replaced by D/$\overline{C}$.

I/O CONTROL SIGNAL GENERATION FOR MEMORY MAPPED I/O

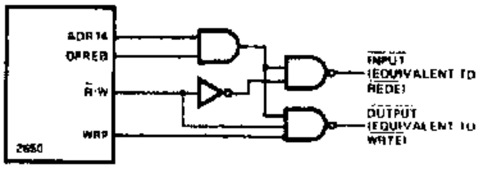


Figure 22

SIXTY-FOUR SINGLE BIT OUTPUTS USING THE 9334

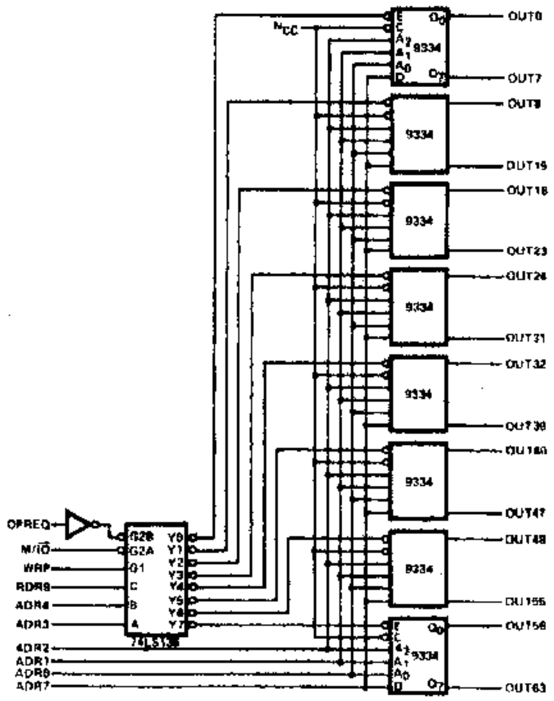


Figure 23

Single Bit Input

One way of doing single bit input uses the techniques described earlier. The address of the bit that is to be tested is loaded into one of the 2650 registers and output to an 8-bit latch using an extended or non-extended write instruction. The latch output is decoded to select the desired bit, which is then applied to the Sense input pin. The 2650 Program Status Word instructions can then be used to test the state of the Sense input and to take appropriate program action.

The technique described above must be used if "indirect" bit addressing is required. If this is not a requirement, a more efficient implementation can be accomplished using the extended read instruction. This technique makes use of the fact that the 2650 automatically tests the contents of a register every time it is used as the destination of an operation. Thus, when the read extended operation reads data from an input port, the condition code bits in the program status word are set to reflect whether the new register contents is positive, negative, or zero.

For the single bit input application, the second byte of the RETE, Rx instruction contains the address of the input bit to be tested. This data is applied to a bank of data selectors to select the addressed bit, which is then applied to the most-significant bit of the data bus, DBUS7. Since this is interpreted as the sign bit, the condition code bits in PSL will be set to reflect whether the bit being tested is a one or a zero. A conditional branch instruction can then be used to affect the desired program action. A hardware implementation for 64 inputs is shown in Figure 24. Note that an address latch is not required for this method.

INPUT PORT DEVICES

Gated Input Ports

The simplest form of an input port is the tri-state gate. Figure 25 illustrates the use of the 8T97 high-speed hex tri-state buffer for gated input ports. The 8T97 is non-inverting, and the tri-state control signals enable the buffers in groups of 4 and groups of 2, so that 8-bit ports can be implemented efficiently.

An effective circuit for systems using 8-gated input ports is the 74251 8-to-1 multiplexer, which has tri-state outputs that can interface directly with the data bus. The advantage of this circuit is that no external address decoding logic is needed. A configuration using gated input ports with the 74251 multiplexer is illustrated in Figure 26.

In addition to these 2 configurations, many other input port configurations are possible using standard TTL or Signetics 8T series logic circuits.

Latching Input Ports

Latching input ports may be required to store data from an external device, which is available only momentarily, before the actual input operation to the microprocessor takes place. This type of input port can be realized by connecting TTL-latch or D-type flip-flop circuits, such as the 7475, 74100, or 74175, to the inputs of a gated input port. As illustrated in Figure 27, by using the Signetics 8T10 Quad D-type flip-flop with tri-state outputs, an 8-bit latching input port can be implemented with only 2 packages. The 8T10 is functionally identical to the 74173.

SIXTY-FOUR SINGLE BIT INPUTS USING THE 74LS251

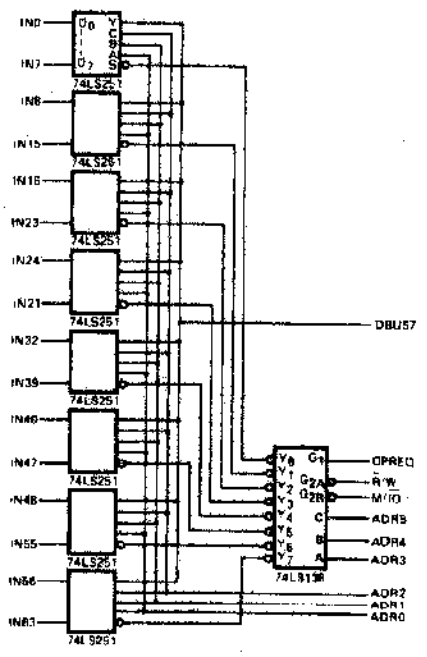
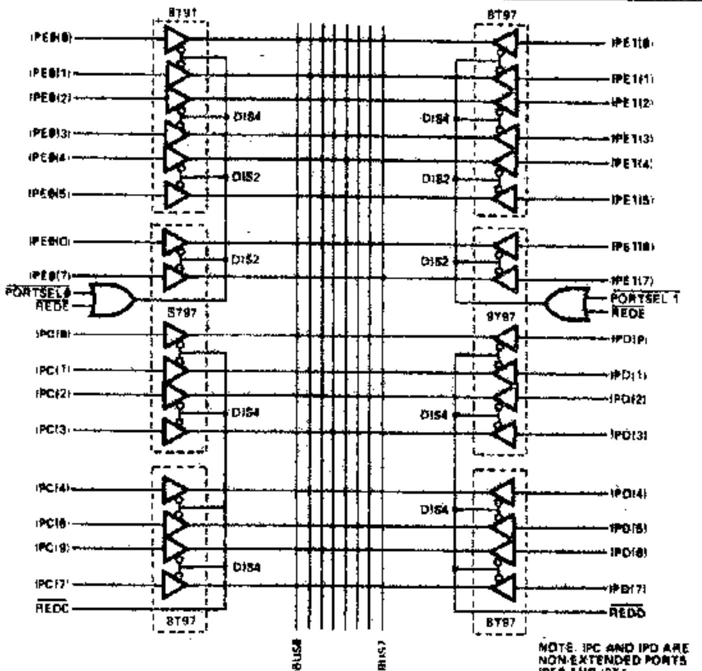


Figure 24



NOTE: IPC AND IPD ARE NON-EXTENDED PORTS. IPE0 AND IPE1 ARE EXTENDED PORTS.

Figure 25

GATED INPUT PORTS WITH 74251 8-to-1 MULTIPLEXERS

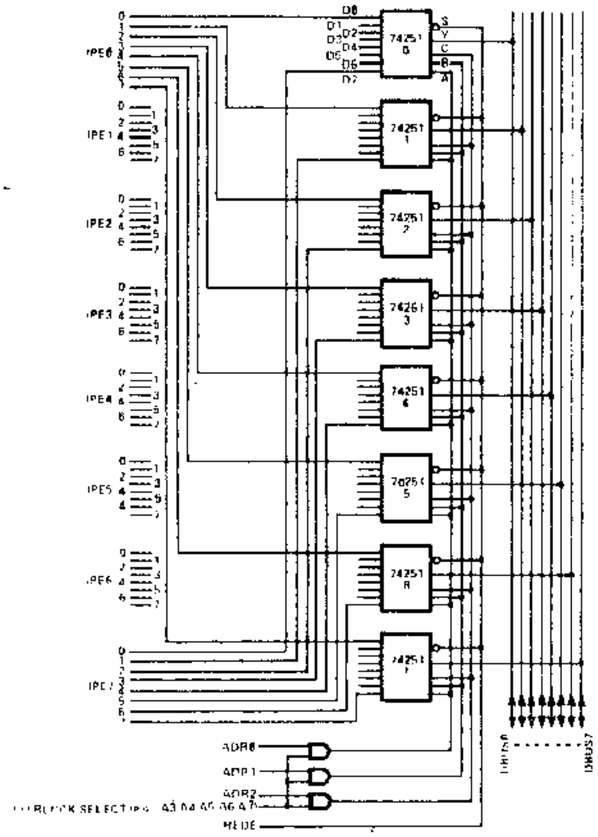


Figure 26

LATCHING INPUT PORTS USING THE 8T10 (74173)

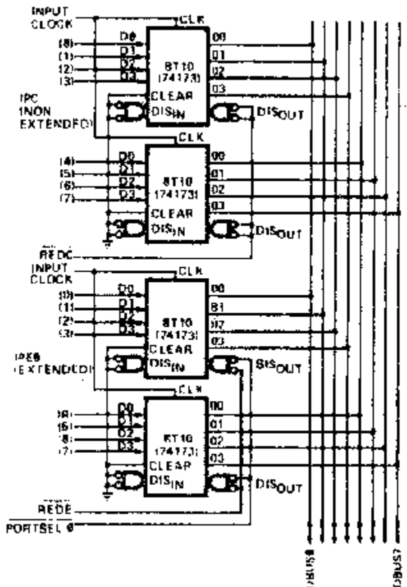


Figure 27

OUTPUT PORTS BUILT WITH STANDARD TTL AND 8T SERIES

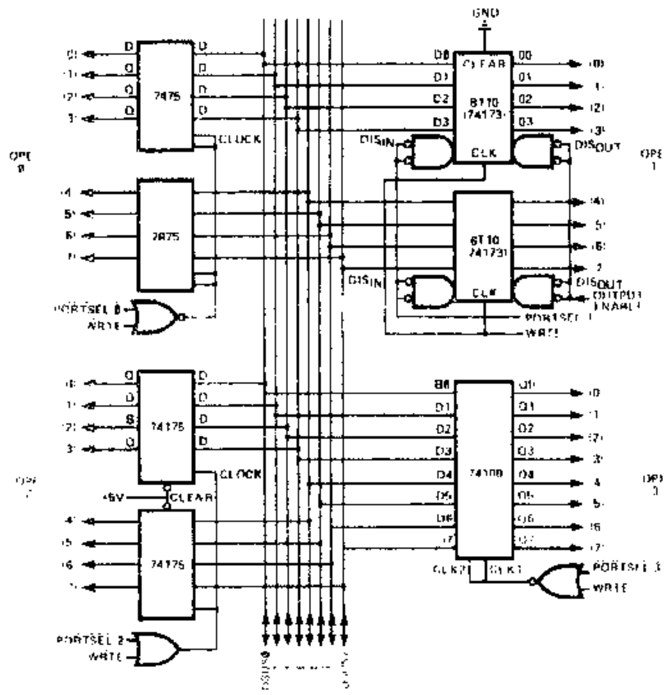


Figure 28

OUTPUT PORT DEVICES

Output ports can be configured with a variety of standard TTL and 8T series flip-flops and registers. Typical circuits include:

9334	Addressable 8-bit latch
7475	Quadruple latch
74100	8-bit latch
74175	Quadruple D-type flip-flop
8T10	Quadruple Q-type flip-flop with tri-state outputs

The 7475 and 74175 both have true and complement outputs. One special feature of the 8T10 is that the outputs may be disabled (placed in a high-impedance output mode) by the device that is connected to this output port. A logic diagram using these circuits for output ports appears in Figure 28.

The 9334 is useful in systems requiring a large number of latched outputs, since a portion of the decoding can be done using the on-chip 3-input decoder. A typical application of this was shown in Figure 23. It is also an efficient circuit for implementing eight 8-bit output ports.

I/O CONFIGURATIONS USING THE 8T31 BIDIRECTIONAL PORT

Functional Description

The 8T31 is an 8-bit bidirectional I/O port consisting of 8 clocked latches with 2 bidirectional I/O buses, each of which has its own control logic. Each bus (A and B) has a read and a write control input, and there is a master enable input for bus B only. The outputs of the latches follow the inputs when the clock is high, and latching will occur when the clock returns low.

The 8T31 is also equipped with a "power-on clear" circuit. If the clock input is held low until the power supply reaches 3.5 volts, the latches will be cleared. There is a logic inversion between bus A and bus B. As a result, when the 8T31 is cleared, bus A will have all logic "1" outputs and bus B all logic "0" outputs.

The control functions of the 8T31 are listed in Table III. A functional block diagram and a symbolic diagram of the 8T31 are illustrated in Figures 29 and 30, respectively.

8T31 FUNCTIONAL BLOCK DIAGRAM

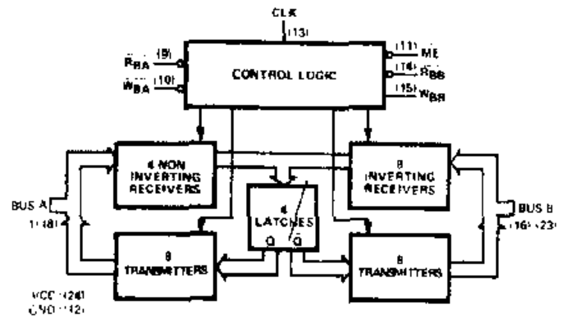


Figure 29

8T31 SYMBOLIC DIAGRAM

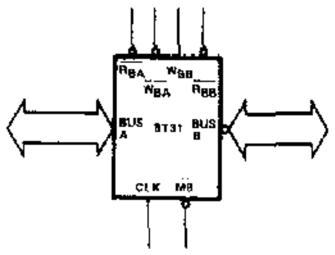


Figure 30

BUS A			
$\overline{R}_{BA}$	$\overline{W}_{BA}$	CLK	BUS A
X	0	1	WRITE (A→latch)
0	1	X	READ (latch →A)
1	1	X	HI-Z (Tri-state)

BUS B					
$\overline{R}_{BB}$	W_{BB}	$\overline{W}_{BA}$	CLK	$\overline{ME}$	BUS B
X	X	X	X	1	HI-Z
1	0	X	X	0	HI-Z
X	1	0	X	0	HI-Z
0	0	X	X	0	READ (latch→B)
X	1	1	1	0	WRITE (B→latch)

Table 3 8T31 CONTROL FUNCTIONS

As shown in Table III, each bus can operate independantly except for the case of writing from both bus A and B. In this case writing from bus A will override any attempt to write from bus B.

8T31 Applications

The control functions of the 8T31 allow it to be used in various microcomputer input output applications. In the I/O system diagram of Figure 31, the 8T31 is used to implement gated input ports, latching input ports, output ports, and a bidirectional data bus driver. All I/O ports can be controlled directly with the device select and $\overline{\text{REDE}}$ and $\overline{\text{WRTE}}$ lines coming from device decoders and I/O control logic.

In applications where interfacing is necessary with peripheral devices that need data transfers in two directions, like digital cassettes and data link communication circuits, the 8T31 can be used as a bidirectional I/O port. In this application, the I/O operation should be requested by interrupt or polling to prevent simultaneous write operations from peripheral and CPU. The bidirectional I/O port concept is illustrated in Figure 32.

Implementing an Eight-Bit Flag Register with the 8T31

In many industrial applications, such as process control, single bit inputs and outputs are used to monitor switches and detectors or to drive relays and lamps. A possible solution for such a flag register would be an eight-bit output port and a memory byte reserved as a flag register in the system's RAM. The setting, resetting, or testing of individual bits with this method of implementing a flag register requires many bytes of program memory. The output port and the memory location reserved as a flag register image must be updated after each bit operation.

The 8T31 can be used to implement a flag register without the use of a memory byte in the system's RAM. No additional hardware is required and the saving in program memory bytes for flag operations is considerable. A logic diagram of this application is given in Figure 33. Listings of basic software to set, reset, and test individual flags for both positive and negative true outputs are given in Figures 34 and 35.

THE 8T31 USED AS A GATED INPUT PORT, LATCHING INPUT PORT, OUTPUT PORT, AND DATA BUS DRIVE

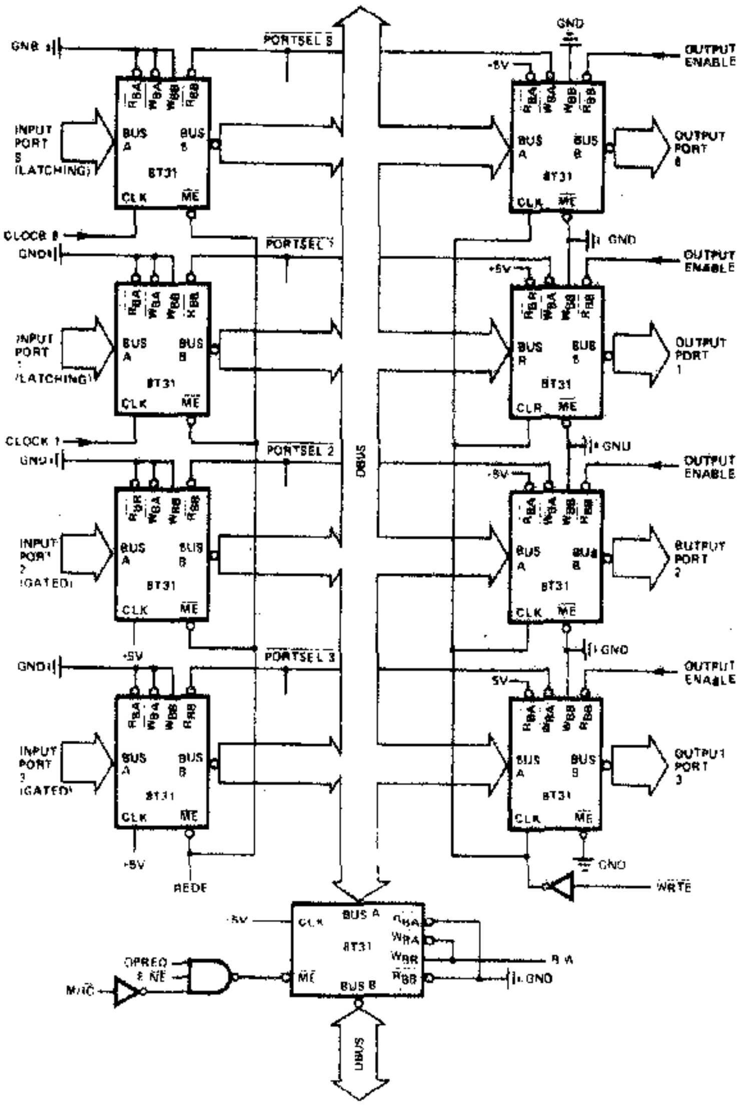


Figure 31