INTERFACE AGE™

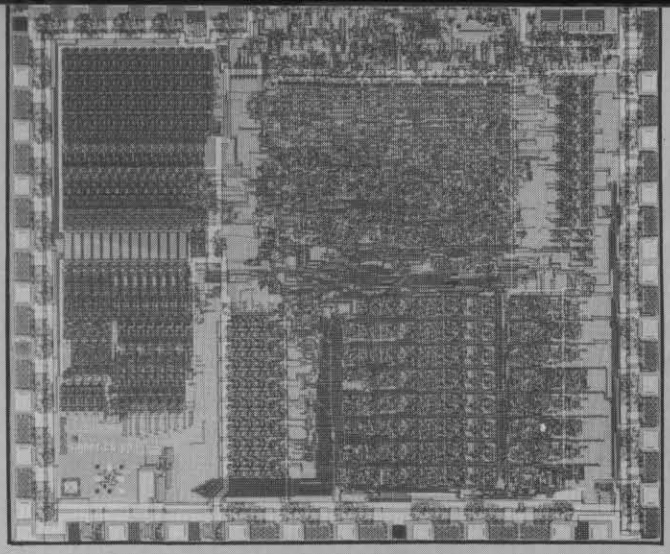
MICROCOMPUTING FOR SMALL BUSINESS AND HOME VOLUME 2, ISSUE 12, NOVEMBER 1977

$1.75
CANADA/MEXICO $2.00
INTERNATIONAL $3.00

How People and Computers Map Space

Also: Computerized Fingerprint Search
General Ledger-III
A Byte of Music
Blockade Game

SPECIAL HARDWARE/SOFTWARE ISSUE



INTERFACE DESIGN WITH

by Alex Goldberger
Signetics, Sunnyvale, California

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 article examines the use of the 2650's 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

The 2650 is equipped with input and output facilities which 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 microprocessor.

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 (ADRO-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.

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 "bottle-neck" in the data flow. The functional block diagram in Figure 1 illustrates the various I/O facilities.

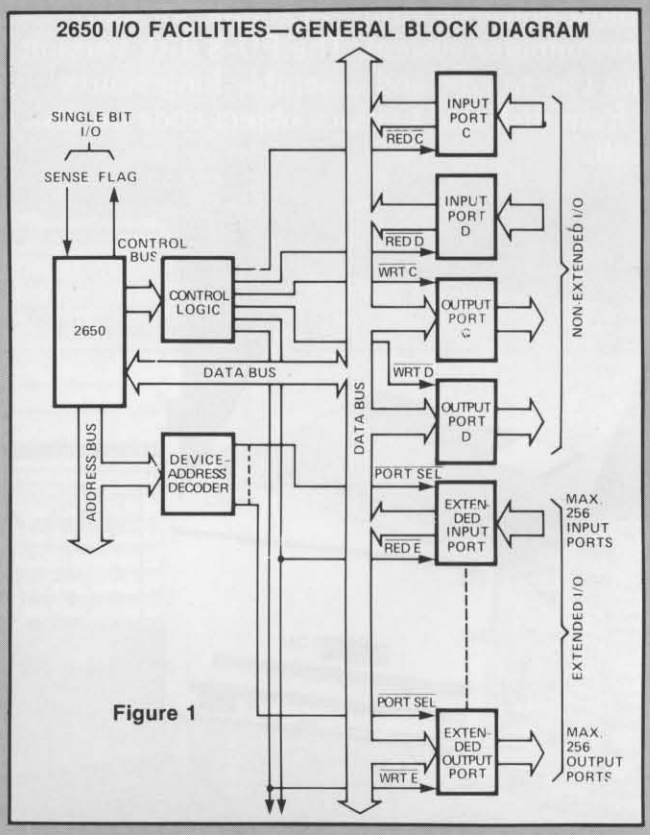


Figure 1

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

SIGNETICS 2650

logical and arithmetic instructions. The memory referencing instructions can take advantage of the flexible addressing modes provided by the system, 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 1 summarizes the state of the 2650 I/O interface signals for the various methods of I/O which are available.

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 give below.

ASYNCHRONOUS SERIAL COMMUNICATIONS PORT: In applications where a serial type of terminal (like a teletypewriter) must be connected to the microcomputer systems, 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.

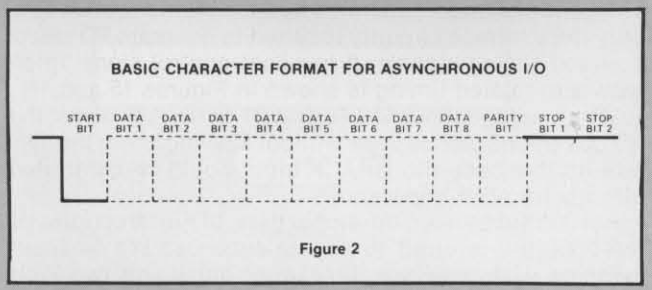


Figure 2

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

Baud rate (bits per second): 110, 150, 300, 600, 1200, 1300, 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 and RS-232 interface or a 20mA current loop interface. The interface hardware is shown in Figure 3.

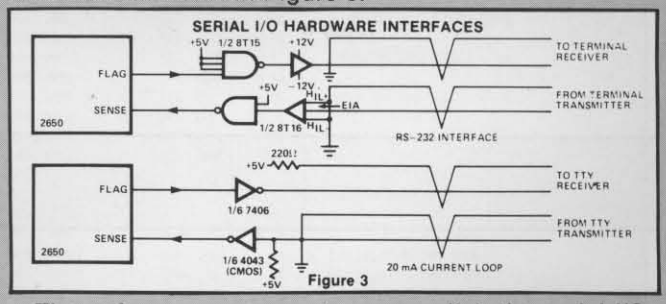


Figure 3

The software necessary to accomplish the serial I/O for a full-duplex line can be divided into 3 parts: 1) 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 1/6 of 1-bit time. This prevents false start-bit recognition caused by line noise. 2) 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. 3) 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.

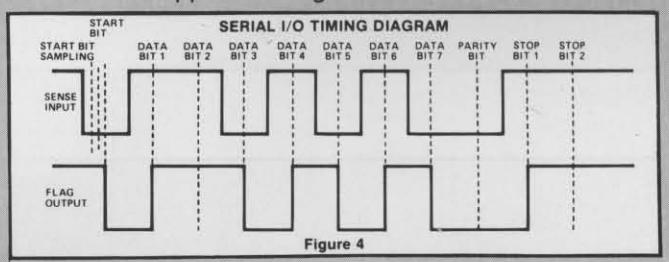


Figure 4

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

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 II

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 III. 8T31 Control Functions

Three examples of the serial I/O routine with different speed and parameters are presented in Figures 5 through 9. The bit and sample delay number (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 11. This table also lists the number of BDRR,RO instructions that are necessary in the "bit delay and echo subroutine" to count cycles for the appropriate baud rate.

The serial I/O routine uses four CPU registers (1 band and RO) and affects seven of the Program Status Word bits; namely, Sense, Flag, Overflow, Carry Interdigit Carry, and the two Condition Code Bits. The program also uses one 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.

DATA SPRING OUTPUT: A typical application for the flag output is a data string output. The advantage for 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, RO to address H'4000'.

PARALLEL INPUT/OUTPUT

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

		NO. BYTES
$\overline{WRTC, RX}$	Write Control	1
$\overline{REDC, RX}$	Read Control	1
$\overline{WRTD, RX}$	Write Data	1
$\overline{REDD, RX}$	Read Data	1
$\overline{WRTE}$, RX DEVA	Write Extended	2
$\overline{REDE}$, 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 $\overline{OPACK}$ 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

FLOWCHART OF THE SERIAL I/O ROUTINE

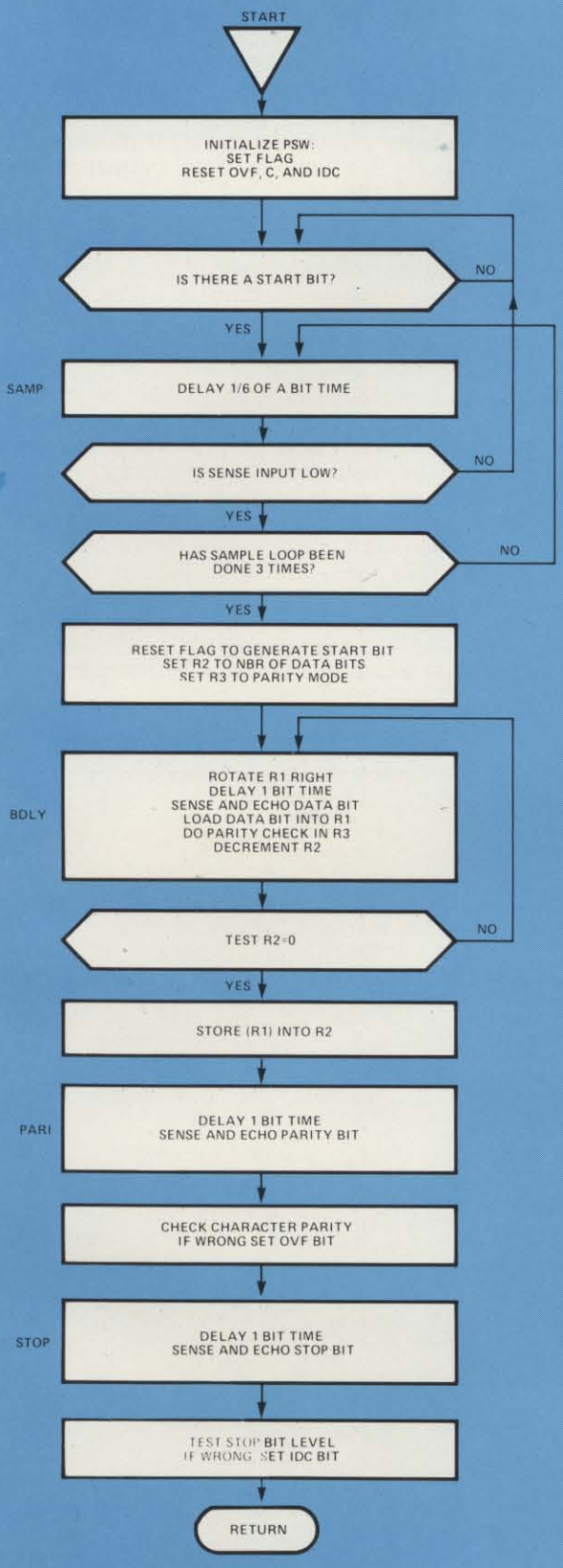


Figure 5

SERIAL I/O PARAMETER DEFINITIONS

```
TWIN ASSEMBLER VER 1.0                                    PAGE 0001

LINE ADDR  OBJECT  E SOURCE

0001                * PD760891
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          DB8  EQU    H'08'     CHARACTER HAS 8 DATA BITS
0042 0000          BP8  EQU    H'00'
0043 0007          DB7  EQU    H'07'     CHARACTER HAS 7 DATA BITS
0044 0040          BP7  EQU    H'40'
0045 0006          DB6  EQU    H'06'     CHARACTER HAS 6 DATA BITS
0046 0020          BP6  EQU    H'20'
0047 0005          DB5  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 00E8          BR01 EQU    H'E8'     BIT DELAY AT 110 BAUD
0053 0069          BR03 EQU    H'69'     BIT DELAY AT 300 BAUD
0054 0060          BR06 EQU    H'60'     BIT DELAY AT 600 BAUD
0055 0053          BR12 EQU    H'53'     BIT DELAY AT 1200 BAUD
0056 0025          BR24 EQU    H'25'     BIT DELAY AT 2400 BAUD
0057                *
0058                * START BIT SAMPLE DELAYS AT 1 MHZ CLOCK FREQUENCY
0059                *
0060 00A5          SD01 EQU    H'A5'     SAMPLE DELAY AT 110 BAUD
0061 003A          SD03 EQU    H'3A'     SAMPLE DELAY AT 300 BAUD
0062 001C          SD06 EQU    H'1C'     SAMPLE DELAY AT 600 BAUD
0063 000C          SD12 EQU    H'0C'     SAMPLE DELAY AT 1200 BAUD
0064 0005          SD24 EQU    H'05'     SAMPLE DELAY AT 2400 BAUD
0065                *
0066                * PARITY MODE
0067                *
0068 0000          EP   EQU    H'00'     EVEN PARITY
0069 0080          OP   EQU    H'80'     ODD PARITY
0070                *
```

Figure 6

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 eight 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 two 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 eight least-significant bits of the address bus, ADR0 through ADR7.

SERIAL I/O ASSEMBLY LISTING—EXAMPLE 1
110 Baud, 7 Data Bits, Even Parity and 1 Stop Bit

```
LINE ADDR  OBJECT  E SOURCE

0073               ********************************************************************
0074               * EXAMPLE 1  FULL DUPLEX (BIT BY BIT ECHO), 110 BAUD,
0075               * 7 DATA BITS, EVEN PARITY AND 1 STOP BIT
0076               *
0077 0000                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 05A5     SAMP LODI,R1 SD01      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 D8?      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     PARI 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 04E8     BDLY LODI,R0 BR01       SET R0 TO BIT DELAY NUMBER
0109 052F F87E           BDRR,R0 $
0110 0531 F87E           BDRR,R0 $
0111 0533 F87E           BDRR,R0 $
0112 0535 F87E           BDRR,R0 $
0113 0537 12             SPSU             TEST DATA BIT 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 DATA BIT INTO R1
0119 0542 23       BIT1 EORZ   R3
0120 0543 C3             STRZ   R3        DO PARITY CHECK
0121 0544 17             RETC,UN
0122               *
0123 0000                END    0

TOTAL ASSEMBLY ERRORS = 0000
```

Figure 7

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 two 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 two 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.

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

EXAMPLE 2
600 Baud, 7 Data Bits, Odd Parity and 2 Stop Bits

```
LINE ADDR  OBJECT  E SOURCE

0073               ********************************************************************
0074               * EXAMPLE 2  FULL DUPLEX (BIT BY BIT ECHO), 600 BAUD,
0075               * 7 DATA BITS, ODD PARITY AND 2 STOP BITS
0076               *
0077 0000                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 051C     SAMP LODI,R1 SD66      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 0708           LODI,R3 OP       SET R3 TO ODD PARITY MODE
0089 0514 0607           LODI,R2 D8?      SET R2 TO NUMBER OF DATA BITS
0090 0516 7440           CPSU   F         GENERATE START BIT
0091 0518 51       BITS RRR,R1
0092 0519 3B1A           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 3B14     PARI BSTR,UN BDLY
0097 0521 9A02           BCFR,N  ST01
0098 0523 7704           PPSL   OVF       IF WRONG PARITY, SET OVF
0099 0525 0700     ST01 LODI,R3 0          CLEAR R3
0100 0527 3B0C           BSTR,UN BDLY
0101 0529 1A02           BCTR,N  ST02     TEST STOP BIT LEVEL
0102 052B 7720           PPSL   IDC       IF WRONG, SET IDC BIT
0103 052D 0700     ST02 LODI,R3 0          CLEAR R3
0104 052F 3B04           BSTR,UN BDLY
0105 0531 16       EXI1 RETC,N             TEST STOP BIT 2 LEVEL
0106 0532 7720           PPSL   IDC       IF WRONG, SET IDC BIT
0107 0534 17       EXI2 RETC,UN
0108               *
0109               ********************************************************************
0110               * BIT DELAY AND ECHO SUBROUTINE
0111               *
0112 0535 04B0     BDLY LODI,R0 BR06       SET R0 TO BIT DELAY NUMBER
0113 0537 F87E           BDRR,R0 $
0114 0539 12             SPSU             TEST DATA BIT LEVEL
0115 053A 1A04           BCTR,N  ONE
0116 053C 7440           CPSU   F         IF LOW, ECHO A ZERO
0117 053E 1B04           BCTR,UN BIT1
0118 0540 7640     ONE   PPSU   F         IF HIGH, ECHO A ONE
0119 0542 6540           IORI,R1 BP7      INSERT DATA BIT INTO R1
0120 0544 23       BIT1 EORZ   R3
0121 0545 C3             STRZ   R3        DO PARITY CHECK
0122 0546 17             RETC,UN
0123               *
0124 0000                END    0

TOTAL ASSEMBLY ERRORS = 0000
```

Figure 8

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.

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 $\overline{\text{WRTE}}$, $\overline{\text{WRTC}}$, and $\overline{\text{WRTD}}$ 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 two 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

EXAMPLE 3
2400 Baud, 8 Data Bits, No Parity and 1 Stop Bit

```
LINE ADDR  OBJECT  E SOURCE

0073               ********************************************************
0074               * EXAMPLE 3  FULL DUPLEX (BIT BY BIT ECHO), 2400 BAUD,
0075               * 8 DATA BITS, NO PARITY AND 1 STOP BIT
0076               *
0077 0000              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 0525     SAMP LODI,R1 SD24      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 0608         LODI,R2 D8         SET R2 TO NUMBER OF DATA BITS
0089 0514 7440         CPSU    F          GENERATE START BIT
0090 0516 51       BITS RRR,R1                .
0091 0517 3B0C         BSTR,UN BDLY       GO TO DELAY AND ECHO ROUTINE
0092 0519 FA7B         BDRR,R2 BITS       TEST FOR NUMBER OF DATA BITS
0093 051B 01           LODZ    R1
0094 051C C2           STRZ    R2         LOAD R2 WITH CHARACTER
0095 051D 0700     STOP LODI,R3 0         CLEAR R3
0096 051F 3B04         BSTR,UN BDLY
0097 0521 16       EXIT RETC,N            TEST STOP BIT LEVEL
0098 0522 7720         PPSL    IDC        IF WRONG, SET IDC BIT
0099 0524 17       EXIT RETC,UN
0100               *
0101               ********************************************************
0102               * BIT DELAY AND ECHO SUBROUTINE
0103               *
0104 0525 0425     BDLY LODI,R0 BR24      SET R0 TO BIT DELAY NUMBER
0105 0527 F87E         BDRR,R0 $
0106 0529 12           SPSU               TEST DATA BIT LEVEL
0107 052A 1A04         BCTR,N  ONE
0108 052C 7440         CPSU    F          IF LOW, ECHO A ZERO
0109 052E 1B04         BCTR,UN BIT1
0110 0530 7640     ONE  PPSU   F          IF HIGH, ECHO A ONE
0111 0532 6580         IORI,R1 BP8        INSERT DATA BIT INTO R1
0112 0534 C3       BIT1 STRZ   R3
0113 0535 17           RETC,UN
0114               *
0115 0000              END     0

 TOTAL ASSEMBLY ERRORS = 0000
```

Figure 9

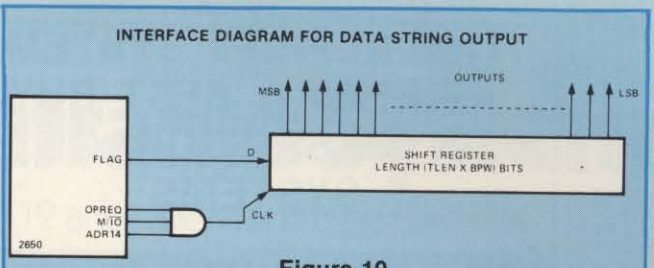


Figure 10.

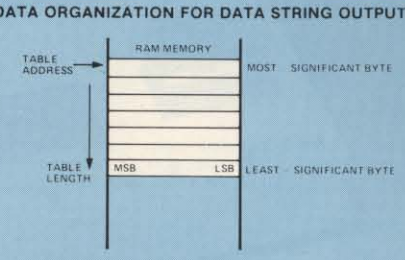


Figure 11.

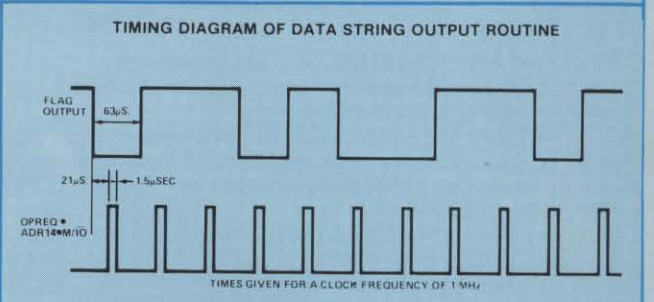


Figure 12.

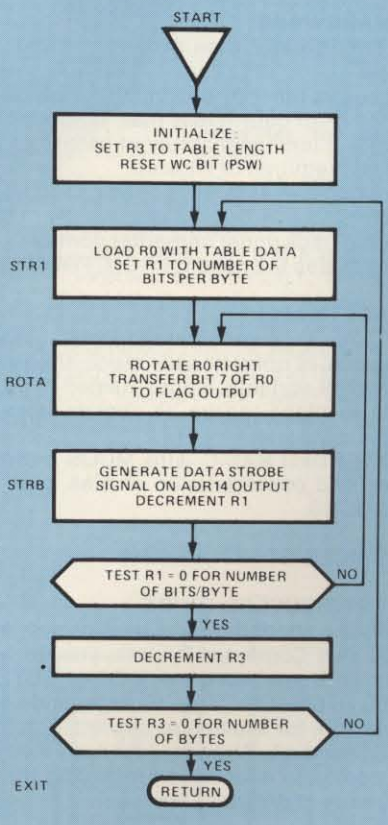


Figure 13.

A_0 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 eight 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 one of the four available registers and outputs in 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 $\overline{\text{WRTD}}$ and $\overline{\text{WRTC}}$ instructions can be used. The address of the output bit is first loaded into one of the 2650 registers. A $\overline{\text{WRTD}}$, Rx instruction is

ASSEMBLY LISTING OF DATA STRING OUTPUT ROUTINE

```
LINE ADDR  OBJECT  E SOURCE

0001               * PD7608B4
0002               ********************************************************************
0003               *                                                                  *
0004               *           **** DATA STRING OUTPUT ROUTINE ****                   *
0005               *                                                                  *
0006               * THIS PROGRAM TRANSFERS THE CONTENTS OF A MEMORY TABLE IN BIT BY  *
0007               * BIT SERIAL FORM TO THE FLAG OUTPUT OF THE 2650                   *
0008               *                                                                  *
0009               * THE TABLE LENGTH AND THE NUMBER OF BITS ARE SOFTWARE PROGRAMMED  *
0010               *                                                                  *
0011               * A DATA STROBE OUTPUT IS GENERATED ON THE ADDRESS 14 OUTPUT       *
0012               *                                                                  *
0013               ********************************************************************
0014               *
0015               * DEFINITIONS OF SYMBOLS
0016               *
0017 0000          R0   EQU    0         PROCESSOR REGISTERS
0018 0001          R1   EQU    1
0019 0002          R2   EQU    2
0020 0003          R3   EQU    3
0021 0080          S    EQU    H'80'     PSU   SENSE
0022 0040          F    EQU    H'40'           FLAG
0023 0008          WC   EQU    H'08'     PSL   1=WITH, 0=WITHOUT CARRY
0024 0002          N    EQU    2         BRANCH COND    NEGATIVE
0025 0003          UN   EQU    3                        UNCONDITIONAL
0026               *
0027 0007          TLEN EQU    H'07'     TABLE LENGTH
0028 0008          BPW  EQU    H'08'     NUMBER OF BITS PER BYTE
0029               *
0030 0000               ORG    H'0600'
0031 0600          TABL RES    TLEN      LOCATION OF TABLE
0032               *
0033               ********************************************************************
0034               *
0035 0607               ORG    H'0500'
0036 0500 0707     STRT LODI,R3 TLEN
0037 0502 7508          CPSL   WC
0038 0504 0F6600   STR1 LODA,R0 TABL,R3   LOAD R0 WITH TABLE DATA
0039 0507 0508          LODI,R1 BPW       SET R1 TO NUMBER OF BITS PER BYTE
0040 0509 50       ROTA RRR,R0
0041 050A 1A06          BCTR,N  ONE       TEST BIT
0042 050C 7440     ZERO CPSU   F          IF ZERO, RESET FLAG
0043 050E 1B04          BCTR,UN STRB
0044               *
0045 0510 4000     ADR  DATA H'40,00
0046               *
0047 0512 7640     ONE  PPSU   F          IF ONE, SET FLAG
0048 0514 CC8510   STRB STRA,R0 *ADR      GENERATE STROBE SIGNAL ON R14
0049 0517 F978          BDRR,R1 ROTA      TEST FOR NUMBER OF BITS
0050 0519 FB69          BDRR,R3 STR1      TEST FOR NUMBER OF BYTES
0051 051B 17       EXIT RETC,UN
0052 0000               END 0

 TOTAL ASSEMBLY ERRORS = 0000
```

Figure 14

Computer Mainframe System

First in the TEI family . . . The MCS-112 and 122 Mainframe Systems.

"The Base on Which to Build"

The cabinet
A heavy duty precision formed cabinet of fine craftsmanship. Completely machined and ready for assembly. The exterior is finished in TEI blue. Vented for most efficient thermal characteristics. Furnished with all necessary hardware.

The motherboard
An S-100 Bus system high quality mother board with 100-pin edge connectors. Compatible with IMSAI, MITS, CROMEMCO, TDL and other S-100 bus configured circuit boards. Plug connections for reset switch. Voltage terminals are screw type to power supply leads. All card guides are provided. 12 slots for MCS-112 model and 22 slots for MCS-122 model.

Edge connectors
High quality edge connectors factory mounted and wave soldered to eliminate this nuisance for you. Completely checked out for shorts or open traces. ALL edge connectors furnished, 12 for the MCS-112 and 22 for the MCS-122. No additional expense when you expand your system.

The power supply
One of a kind . . . using a constant voltage transformer (CVT) with a very high immunity to input line noise . . . greater than 100 db rejection. Line regulation better than ± 1% from an input of 95 to 140 Volt AC at full load to 85 to 140 Volt AC at three quarter load. Designed to meet UL-478 specifications (EDP SPECS). Individual fusing on all input and output voltage lines. See specifications below for power ratings.

The cooling system
A 115 CFM muffin fan with a commercial grade washable filter will provide clean airflow over all circuitry.

The wiring
All wiring is color coded and ALL is precut to length with connecting lugs factory machine applied. Soldering is held to an absolute minimum.

The front panel
The front panel is blank except for an indicating AC switch and a reset switch. However, the chassis and mother board are designed so that you may remove the front panel and insert an IMSAI or equivalent front panel. Soon to be available will be our "VIRTUAL OPERATING CONSOLE" especially designed to complement our Mainframe Systems.

Specifications	MCS-112	MCS-122
Dimensions	17¼ W x 12 D x 7¼ H	17¼ W x 19½ D x 7¼ H
Power +8 volt DC	17 amps	30 amps
Power ±16 volt DC	9 amps	4 amps

NOW . . . TEI puts it all together for you. Mainframe systems (6, 12 and 22-slot) . . . Floppy and mini-floppy disc drive systems (single, double and triple with dual density) . . . a Z-80 CPU with addressable "jump to" and autostart capability (also variable speed control) . . . 8K and 16K RAM . . . 16 Channel A/D and D/A converter . . . 3 serial + 3 parallel multiple I/O . . . and other supporting boards . . . and our newest item, the PROCESSOR TERMINAL — A CRT, keyboard, mini-floppy disc and 12-slot mainframe with a 8080A CPU – all housed in one quality aluminum case. All of these fine products at prices you will like. **Watch for them all.**

Contact your local TEI dealer or if you are not near one of our dealers, write or call CMC MARKETING CORP direct for more information.

(DEALER INQUIRIES INVITED)

PLEASE SEND ME:

☐ MCS-112 Kit @ 395.00 ☐ MCS-112 Assembled @ 445.00
☐ MCS-122 Kit @ 495.00 ☐ MCS-122 Assembled @ 575.00

Texas residents add 5% Sales Tax

I Enclose Check ☐ or Money Order ☐

CMC MARKETING CORP

7231 Fondren Rd, Houston, TX 77036 Telephone (713) 774-9526

TEI INC.

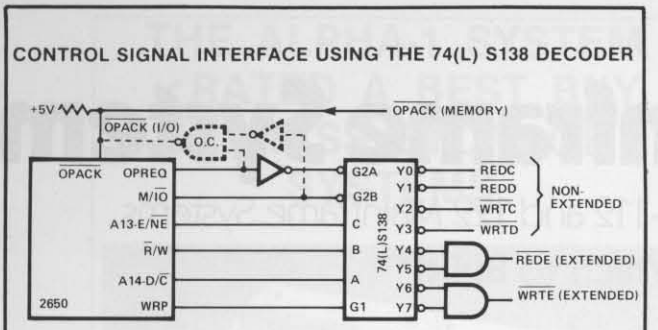


Figure 15.

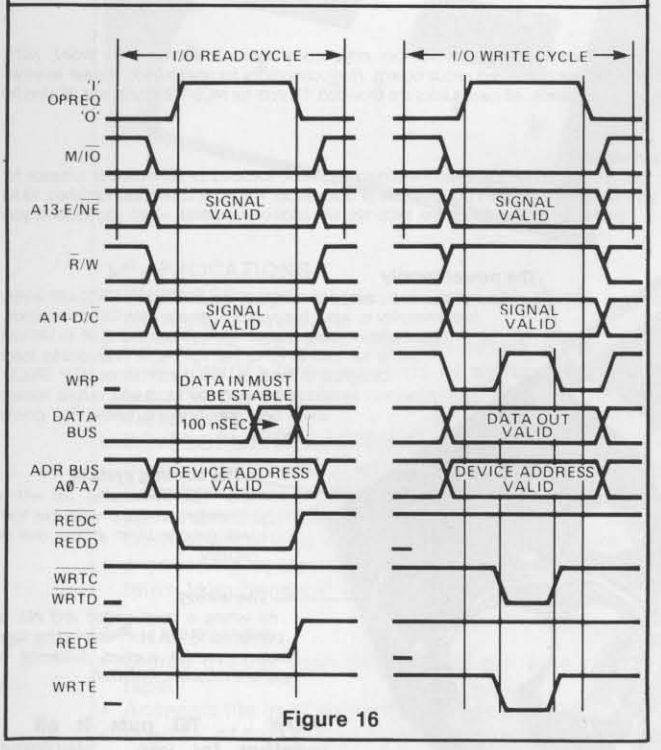


Figure 16

then issued if the bit is to be set, and a $\overline{\text{WRTC}}$, Rx instruction is issued if the bit is to be cleared. The bit select is output on the data bus, and the D/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/C.

The 8T31 can be used to implement a flag register without the use of a memory byte in RAM. No additional hardware required and memory savings are considerable.

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.

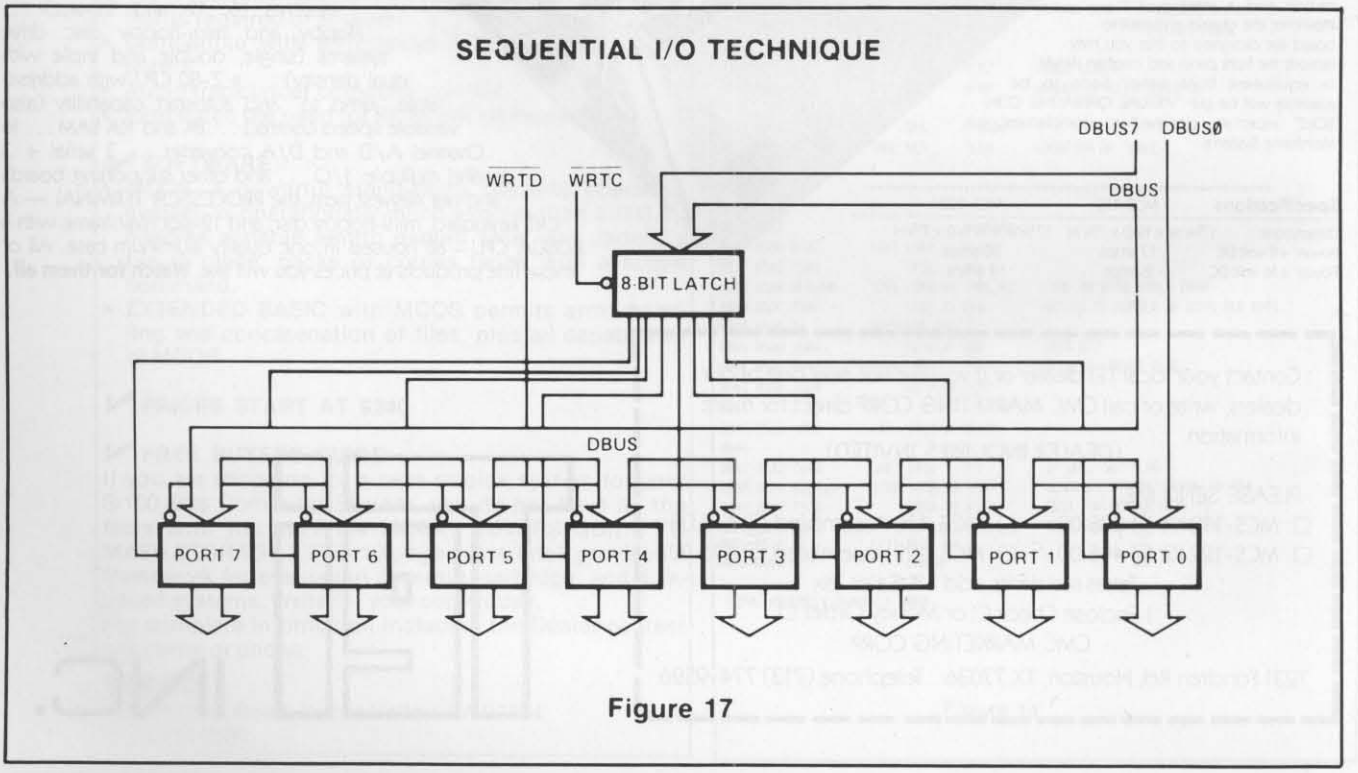


Figure 17

New UVS-11E EPROM Erasing System

Performance and Reliability for $59.50

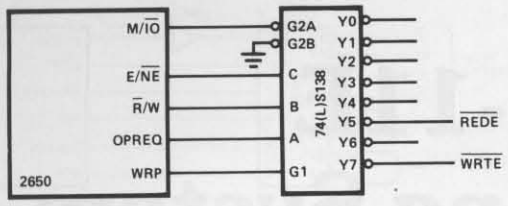
Now available . . . the newest member of UVP's growing family of quality EPROM Erasing Lamps. The UVS-11E Short Wave UV Lamp was designed specifically for the small systems user and computer hobbyist. It's compact, easy-to-use, and will erase up to 4 chips at one time. It even features a special safety interlock system for complete safety.

This is the first UV erasing system to offer simple operation and foolproof safety features at an affordable price. Like all UVP products, the UVS-11E is quality-built and backed by 45 years of UV technology.

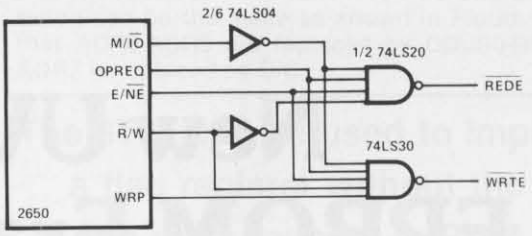
Order now from your local authorized UVP stocking dealer. Or write today for more information and name of nearest dealer.

ULTRA-VIOLET PRODUCTS, INC.
5100 Walnut Grove Avenue, San Gabriel, CA 91776 U.S.A.

SIMPLIFIED CONTROL LOGIC WHEN USING EXTENDED I/O ONLY



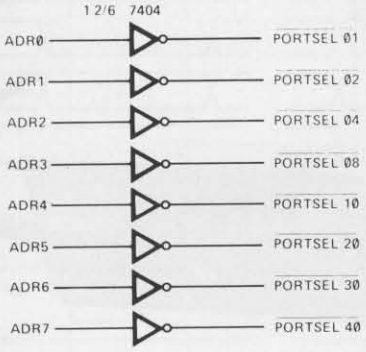
(A) Using 1-of-8 Decoder



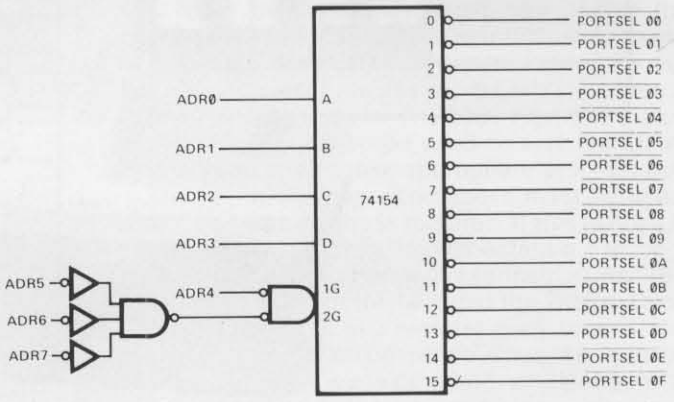
(B) Using Logic Gates

Figure 18

SOME POSSIBLE TECHNIQUES FOR DEVICE ADDRESS DECODING

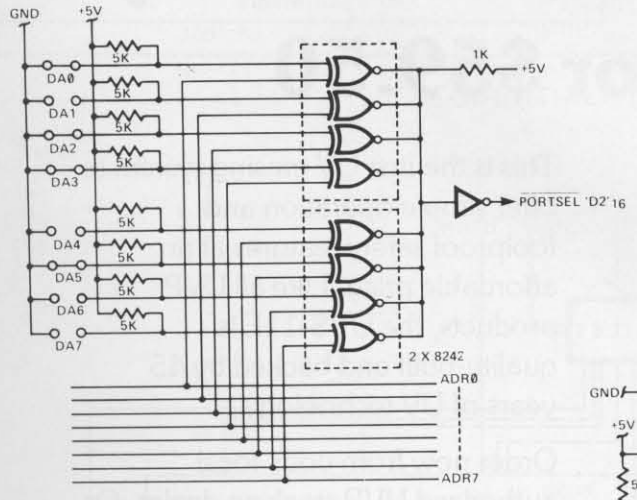


(A) Each address line selects one device (maximum of eight)

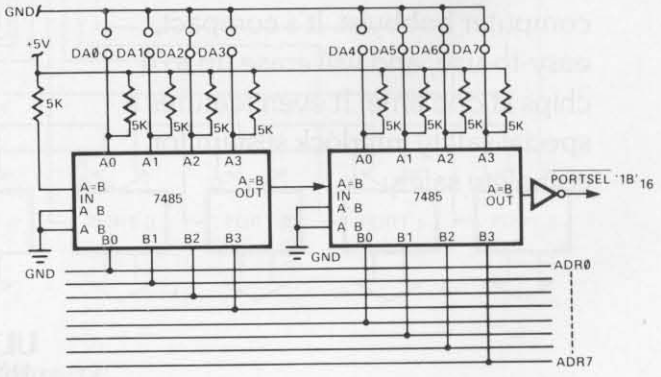


(B) Using a 1-out-of-16 decoder

Figure 19



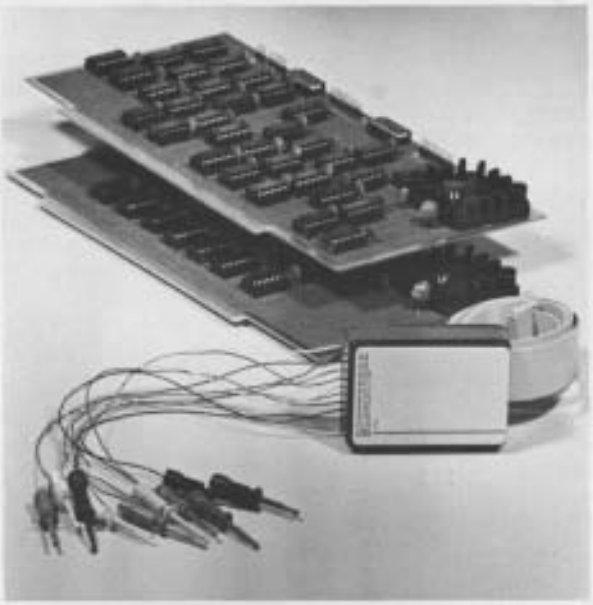
(A) Using Exclusive-OR Gates



(B) Using Comparators

Figure 20

24 channel Logic Analzer plugs into your S-100 Bus



24 Channel **LOGIC ANALYZER**, complete with 2 cards and 3 sets of probes (only one set shown).

Features

- 24 channels with 256 samples each.
- Display of disassembled program flow.
- Dual mode operation – external mode analyses any external logic system. Internal mode monitors users data and address bus.
- Selectable trigger point anywhere in the 256 samples.
- 0-16 bit trigger word format or external qualifier.
- 10MHz sample rate (50ns min. pulse width)
- Synchronous clock sample with coincident or delayed clock mode.
- User defined reference memory.
- Displays and system control through keyboard entry.
- TTL Logic level compatible (15 pf and 15 μa typical input loading).
- Includes annotated source listing.

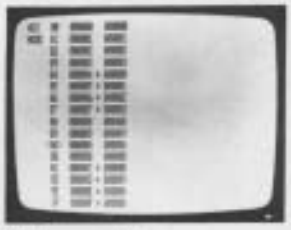
Display of disassembled program flow.

The DATALYZER

The Databyte Logic Analyzer (DATALYZER) is a convenient, flexible, high quality device. Efficient engineering has allowed a combination of features previously available in only the most expensive units.

Designed to plug easily into your S-100 Bus, the DATALYZER is a complete system — for only $495. Display of disassembled program flow is a standard feature, not an extra. And the low price includes 30 logic probes, so you can hook up immediately, without additional expense.

The DATALYZER is available in kit form ($495), and as a fully assembled device on two PCB's ($595). Four-week delivery, a substantial warranty, and the Databyte, Inc. commitment to service make the DATALYZER a worthwhile investment. Begin debugging by sending the coupon now.



Displays in Hex

Displays in Binary

Please send me the 24 Channel **LOGIC ANALYZER**

☐ Kit – (manual included) $495.00 (Wis. res. add 4%)

☐ Assembled and Tested (manual included) $595.00

☐ Operators' manual only $7.50

Delivery of all items in four weeks to:

Name _____

Address _____

City _____ State _____ Zip _____

Telephone _____

Payment Enclosed: ☐ Check ☐ Money Order

☐ BankAmericard ☐ Master Charge Exp. Date _____

Number _____

Signature _____

Databyte, Inc.

P.O. Box 14
7433 Hubbard Avenue
Middleton, Wisconsin 53562
Tel: (608) 831-7666

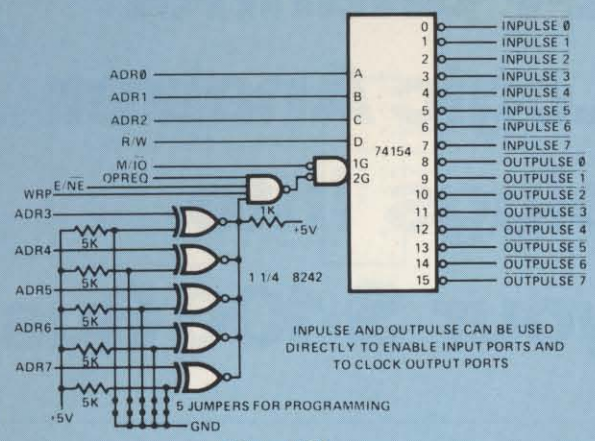


Figure 21.

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

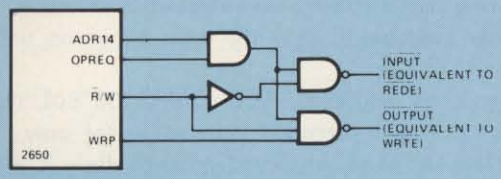


Figure 22.

SIXTY-FOUR SINGLE BIT OUTPUTS USING THE 9334

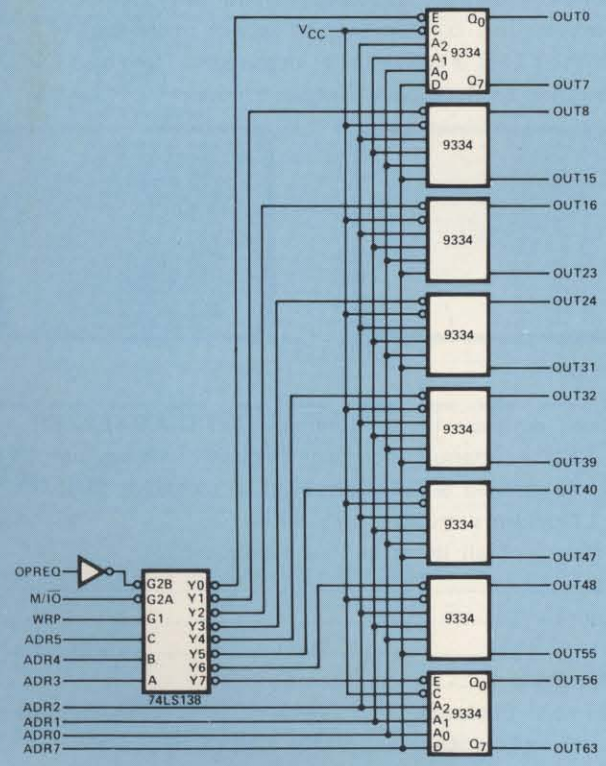


Figure 23.

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 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 four and groups of two, 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 two 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 two packages. The 8T10 is functionally identical to the 74173.

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 D-type flip-flop with tri-state ouputs

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

The 8T31 is an 8-bit bidirectional I/O port consisting of eight clocked latches with two 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 3.5V, 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 111. A functional block diagram and a symbolic diagram of the 8T31 are illustrated in Figures 29 and 30, respectively. As shown in Table 111, each bus can operate independently except for the care of writing from both bus A and B. In this case writing from bus A will override any attempt to write from bus B.

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

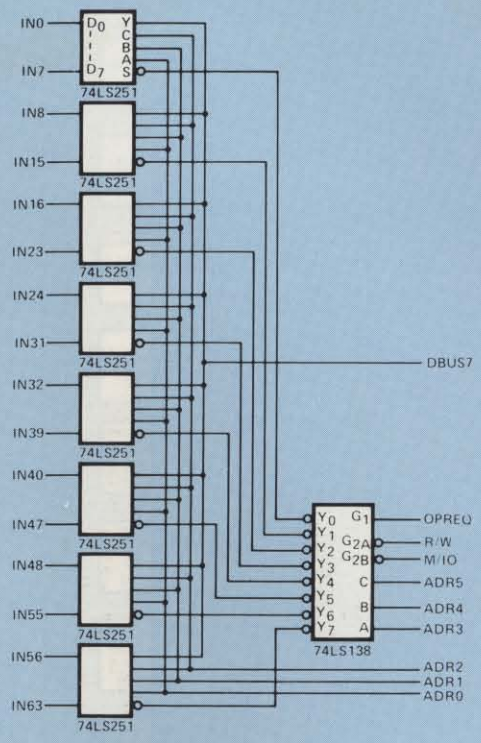


Figure 24

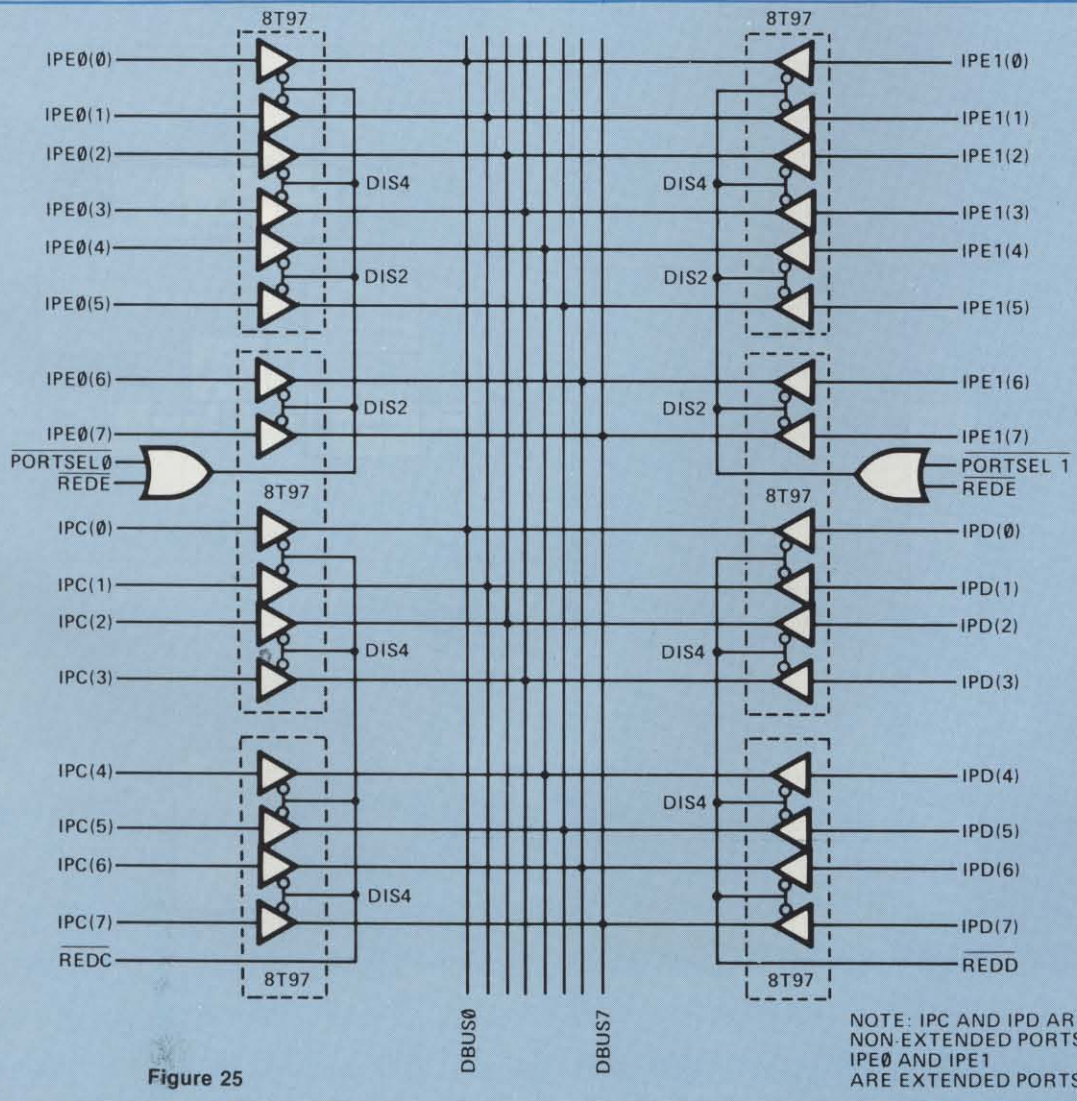


Figure 25

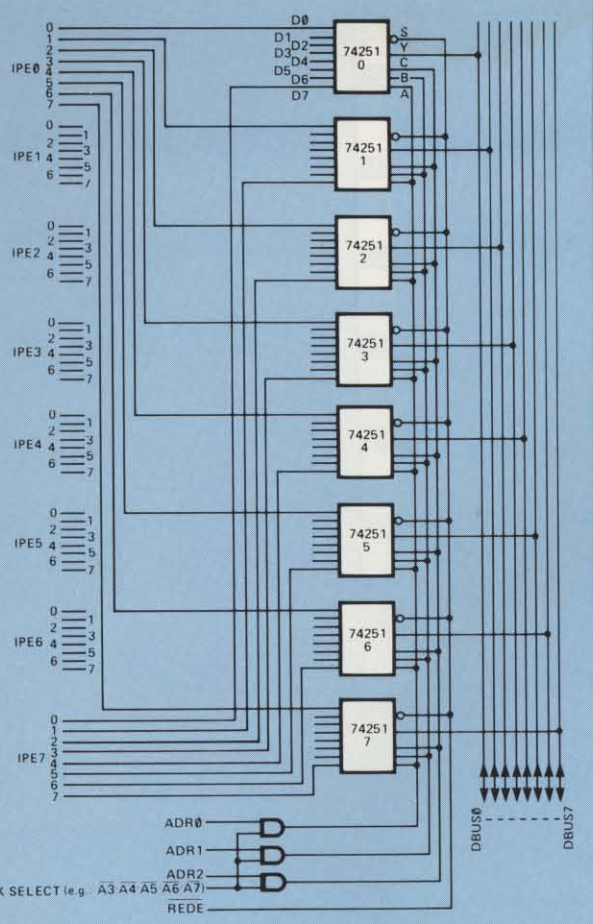


Figure 26

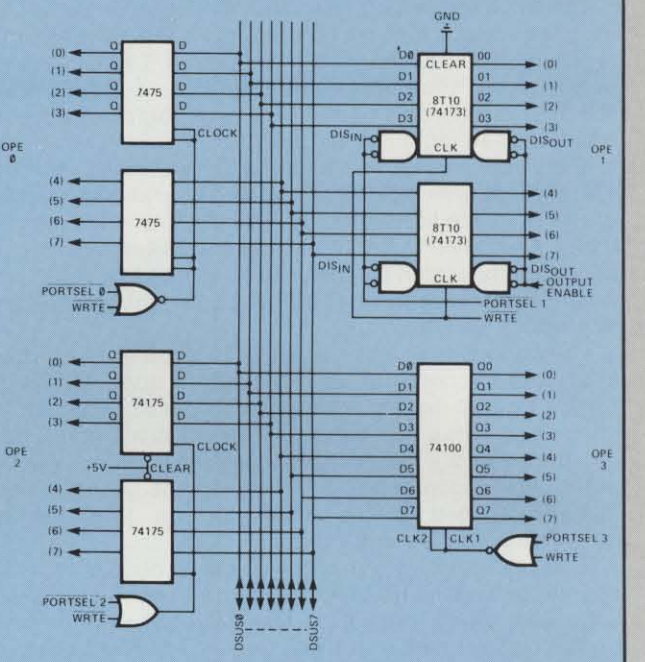


Figure 28

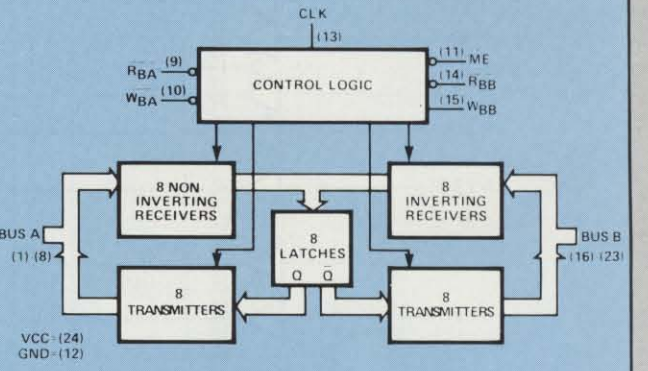


Figure 29

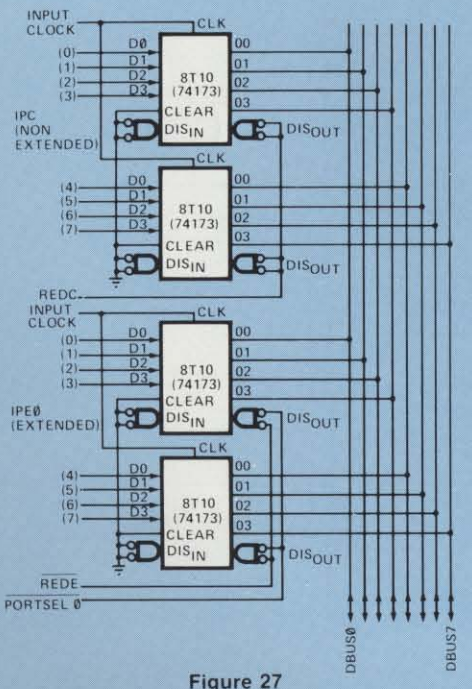


Figure 27

8T31 SYMBOLIC DIAGRAM

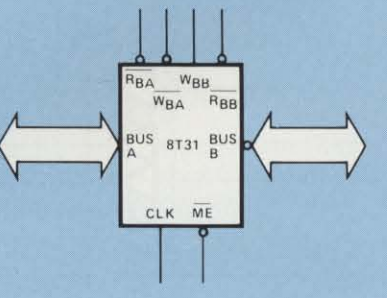


Figure 30

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

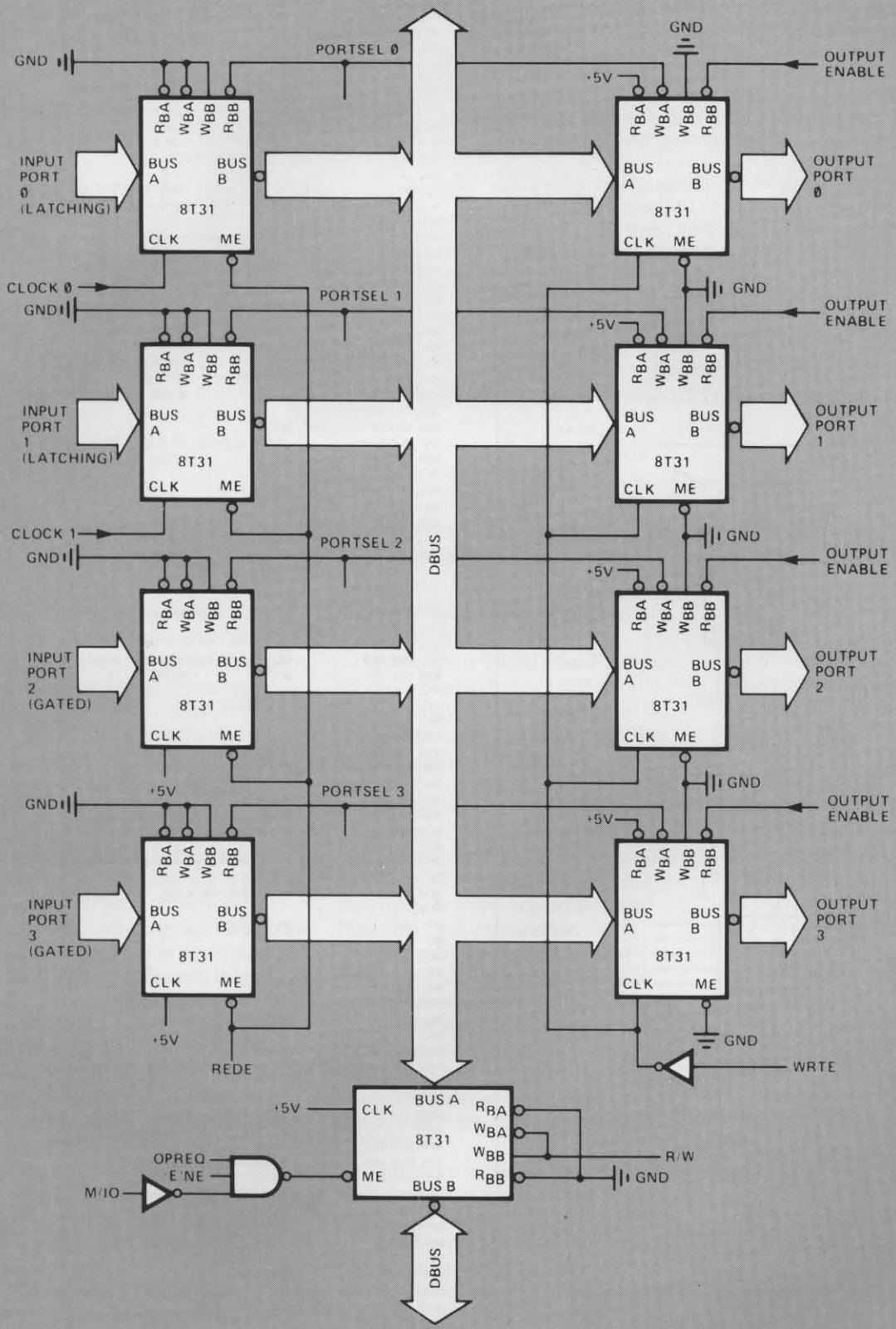


Figure 31

write operations from peripheral and CPU. The bidirectional I/O port concept is illustrated in Figure 32.

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 an 8-bit flag register would be an 8-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 BIDIRECTIONAL I/O PORT

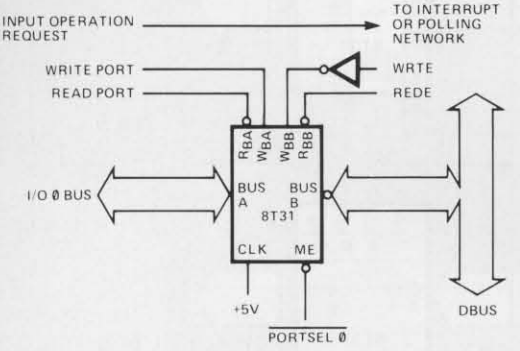


Figure 32

A FLAG REGISTER IMPLEMENTED WITH THE 8T31

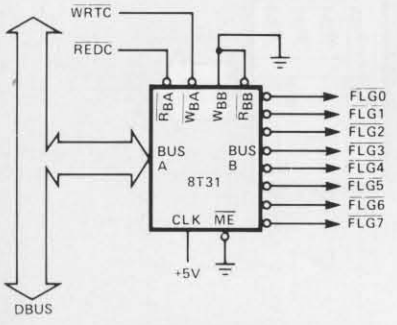


Figure 33

```
TWIN ASSEMBLER VER 1.0                                  PAGE 0001

LINE ADDR  OBJECT  E SOURCE

0001               * P0760890
0002               ************************************************
0003               *                                              *
0004               *     **** FLAG MANIPULATION EXAMPLES ****     *
0005               *                                              *
0006               * THIS LISTING GIVES SOME EXAMPLES HOW TO SET, RESET   *
0007               * AND TEST INDIVIDUAL BITS OF AN EXTERNAL FLAG REGISTER *
0008               * BUILT WITH THE 8T31 BIDIRECTIONAL I/O PORT   *
0009               * INSTRUCTIONS ARE GIVEN FOR BOTH ACTIVE 'HIGH' AND    *
0010               * ACTIVE 'LOW' OUTPUTS                         *
0011               *                                              *
0012               ************************************************
0013               *
0014               * DEFINITIONS OF SYMBOLS
0015               *
0016 0000          R0   EQU   0        PROCESSOR REGISTERS
0017 0001          R1   EQU   1
0018 0002          R2   EQU   2
0019 0003          R3   EQU   3
0020 0000          Z    EQU   0        BRANCH COND    ZERO
0021 0003          UN   EQU   3                       UNCONDITIONAL
0022 0000          AL   EQU   0                       ALL BITS ARE 1
0023               *
0024 0001          FLG0 EQU   H'01'    FLAG 0
0025 0002          FLG1 EQU   H'02'    FLAG 1
0026 0004          FLG2 EQU   H'04'    FLAG 2
0027 0008          FLG3 EQU   H'08'    FLAG 3
0028 0010          FLG4 EQU   H'10'    FLAG 4
0029 0020          FLG5 EQU   H'20'    FLAG 5
0030 0040          FLG6 EQU   H'40'    FLAG 6
0031 0080          FLG7 EQU   H'80'    FLAG 7
0032               *
0033 0600          ONE  EQU   H'0600'  DUMMY ADDRESS OF ROUTINE 'ONE'
0034 0650          ONES EQU   H'0650'  DUMMY ADDRESS OF ROUTINE 'ONES'
0035               *
0036               ************************************************
0037               *
0038               * **INSTRUCTIONS FOR ACTIVE 'LOW' OUTPUTS**
0039               *
0040 0500                ORG   H'0500'
0041               *
0042               *    SET FLAG(S)
0043               *
0044 0500 30       SNFG REDC,R0          LOAD FLAG REGISTER IN R0
0045 0501 6404           IORI,R0 FLG2    SET FLAG 2
0046 0503 B0             WRTC,R0         RESTORE FLAG REGISTER
0047               *
0048 0504 30       SNFS REDC,R0
0049 0505 6460           IORI,R0 FLG5+FLG6 SET FLAGS 5 AND 6
0050 0507 B0             WRTC,R0         RESTORE
0051               *
0052               *    RESET FLAG(S)
0053               *
0054 0508 30       RNFG REDC,R0
0055 0509 44FB           ANDI,R0 H'FF'-FLG2 RESET FLAG 2
0056 050B B0             WRTC,R0         RESTORE
```

Figure 34

```
TWIN ASSEMBLER VER 1.0                                  PAGE 0002
LINE ADDR  OBJECT  E SOURCE
0058 050C 30       RNFS REDC,R0
0059 050D 449F           ANDI,R0 H'FF'-FLG5-FLG6 RESET FLAGS 5 AND 6
0060 050F B0             WRTC,R0         RESTORE
0061               *
0062               *    TEST FLAG(S)
0063               *
0064 0510 30       TNFG REDC,R0
0065 0511 F404           TMI,R0  FLG2    TEST FLAG 2
0066 0513 1C0600         BCTA,AL ONE     BRANCH IF ONE
0067               *
0068 0516 30       TNFS REDC,R0
0069 0517 F460           TMI,R0  FLG5+FLG6 TEST FLAGS 5 AND 6
0070 0519 1C0650         BCTA,AL ONES    BRANCH IF BOTH ARE ONE
0071               *
0072               ************************************************
0073               *
0074               * **INSTRUCTIONS FOR ACTIVE 'HIGH' OUTPUTS**
0075               *
0076 051C                ORG   H'0550
0077               *
0078               *    SET FLAG(S)
0079               *
0080 0550 30       SPFG REDC,R0
0081 0551 44FB           ANDI,R0 H'FF'-FLG2 SET FLAG 2
0082 0553 B0             WRTC,R0         RESTORE
0083               *
0084 0554 30       SPFS REDC,R0
0085 0555 44ED           ANDI,R0 H'FF'-FLG1-FLG4 SET FLAGS 1 AND 4
0086 0557 B0             WRTC,R0         RESTORE
0087               *
0088               *    RESET FLAG(S)
0089               *
0090 0558 30       RPFG REDC,R0
0091 0559 6404           IORI,R0 FLG2    RESET FLAG 2
0092 055B B0             WRTC,R0         RESTORE
0093               *
0094 055C 30       RPFS REDC,R0
0095 055D 6412           IORI,R0 FLG1+FLG4 SET FLAGS 1 AND 4
0096 055F B0             WRTC,R0         RESTORE
0097               *
0098               *    TEST FLAG(S)
0099               *
0100 0560 30       TPFG REDC,R0
0101 0561 F404           TMI,R0  FLG2    TEST FLAG 2
0102 0563 9C0600         BCFA,AL ONE     BRANCH IF ONE
0103               *
0104 0566 30       TPFS REDC,R0
0105 0567 F412           TMI,R0  FLG1+FLG4 TEST FLAGS 1 AND 4
0106 0569 9C0650         BCFA,AL ONES    BRANCH IF BOTH ARE ONE
0107               *
0108 0000                END     0

TOTAL ASSEMBLY ERRORS = 0000
```

Figure 35