COMPUTE

the Club Of Microprocessor Programmers, Users, and Technical Experts

Georgia Marszalek, Editor • David Graves, Editor

Sponsored by National Semiconductor Corp., Santa Clara, Ca. 95051 **Vol. 2, No. 9, September, 1976**

Increasing Throughput for IMP-16 Serial Input/Output

Application Note 160, by Keith Winter, National Semiconductor

INTRODUCTION

A method of increasing the throughput for serial transfer of input/output data between the IMP-16 microprocessor and peripheral devices is described in this application note. Obtaining maximum throughput is of great importance for a high-performance microprocessor such as the National Semiconductor IMP-16. Until recently, however, the best method for generating serial input/output transfers with microprocessors had been to use a program-control method. This method requires the use of a flag output and a sense input to send and receive data. The bit-time generation is accomplished under program control by loading a RAM counter with a fixed constant determined by the baud rate desired and then decrementing this counter until it reaches zero. At the completion of the count, the next bit is sent, and the process is repeated until all bits of the character have been sent. The same method is used when receiving a character. This method has two serious drawbacks: inefficiency and possible loss of information.

The program-control method is inefficient with respect to throughput since the CPU is dedicated to bit-time generation and is unable to perform any other functions. This amounts to a substantial loss of CPU time if the baud rate is low or if there is a large amount of data that must be transferred over a serial line. In addition to the bit serialization (parallel-to-serial conversion) of the character, the CPU must compute parity, if any, and perform the function of adding and deleting start and stop bits. Figure 1 is a graph of average instruction execution times versus character transmittal times. As evident from the graph, at low baud rates, a large quantity of CPU time is consumed during bit serialization. At 110 baud, for example, an average of more than 12,000 instruction times are used to produce one character. Refer to table 1 for exact times and numbers of instructions per character.

The possible loss of characters is a problem if the CPU were to be interrupted in the middle of the data stream. This is evident in the IMP-16P system at baud rates of 300 and above when control panel interrupts are enabled. The panel interrupts every 100 milliseconds, and at 300 baud this is once every third or fourth character. The resultant interrupt service routine causes program control to be transferred out of the bit-time generation routine. The result is invalid bit sampling and garbled characters. An identical situation exists for general user interrupts that cannot be tolerated during serial input/output timing.

Obviously, a device that performs all of the functions previously mentioned and removes them from CPU responsibility is desirable. Such a device, available as a single integrated circuit, is the UAR/T (Universal Asynchronous Receiver/Transmitter).

SUMMARY OF UAR/T OPERATION

The MM5303 UAR/T is an MOS/LSI device that takes parallel 8-bit data from the CPU and converts it to a serial data stream for transmission. On reception, a serial bit stream is converted to a parallel character that the CPU can take in a single instruction. The UAR/T has provisions to select externally even parity, odd parity, or no parity, 5-, 6-, 7-, or 8-bit data lengths, and baud rate as a function of the external clock frequency. Since the transmit and receive clocks are separate, the UAR/T can send and receive simultaneously at different baud

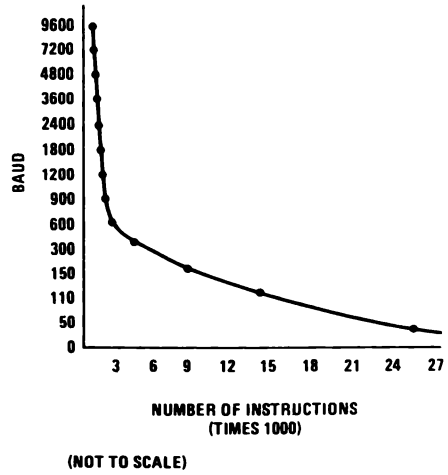


Figure 1. Baud Rate Vs. Number of Instructions

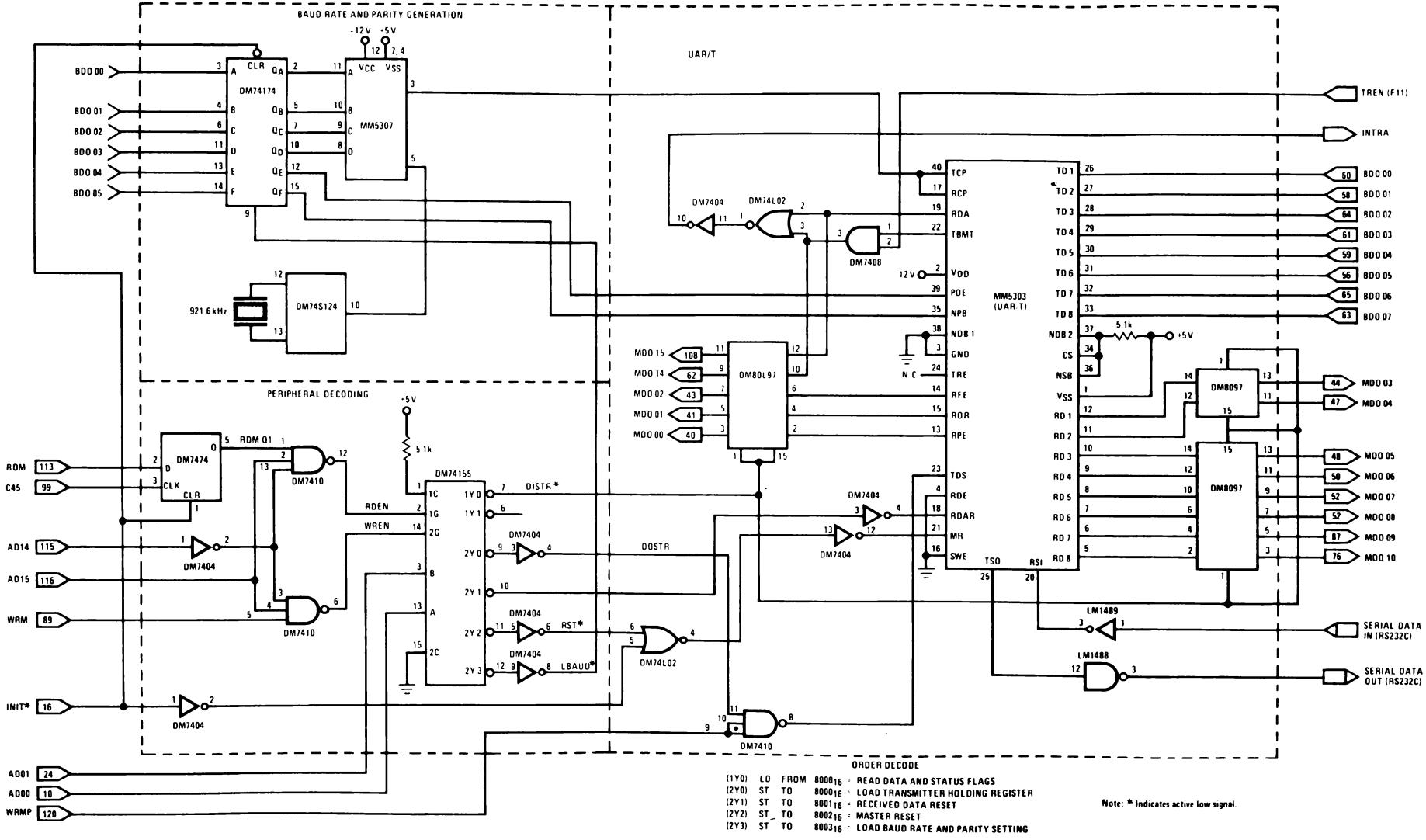


Figure 2. UAR/T Interfacing

Baud Rate	Characters per Second	Time per Character	Approximate Number of Instructions Executed During Character Transmission*
50	5	200ms	24,875
110	10	100ms	12,438
150	15	66ms	8,209
300	30	33ms	4,104
600	60	16ms	1,990
900	90	11ms	1,368
1200	120	8.33ms	1,036
1800	180	5.5ms	691
2400	240	4.16ms	518
3600	360	2.77ms	345
4800	480	2.08ms	258
7200	720	1.38ms	173
9600	960	1.04ms	129

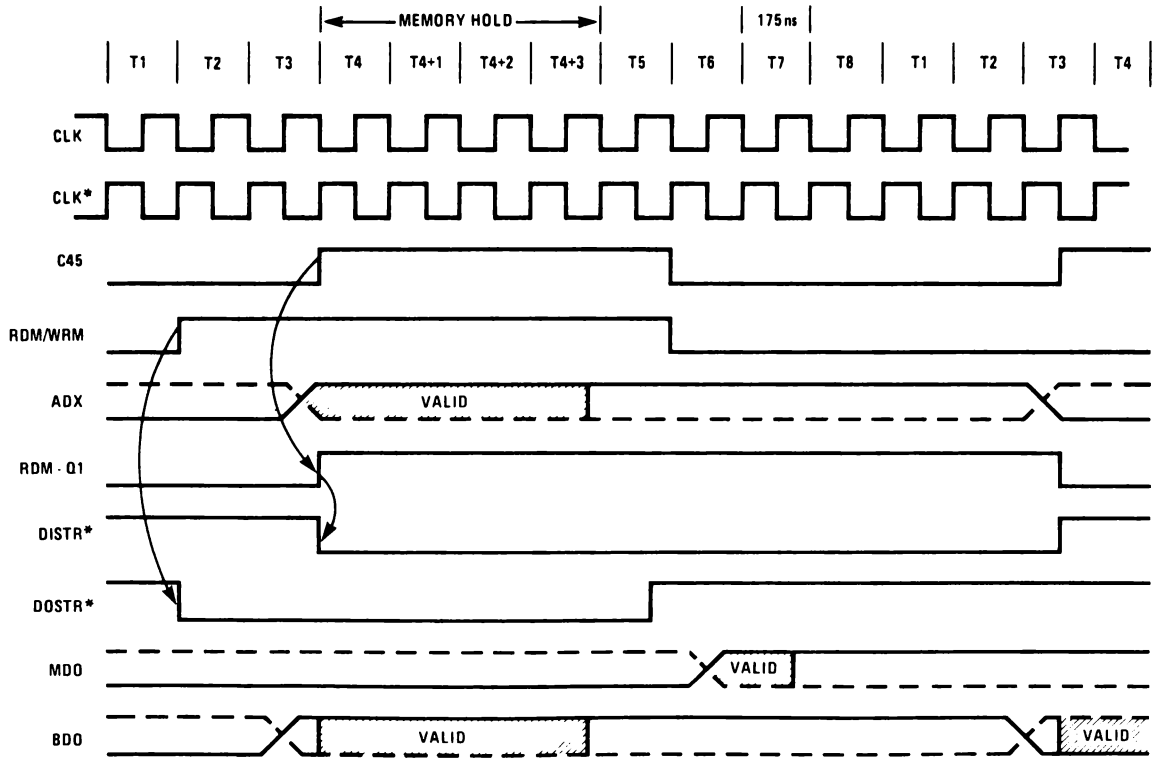
*Using average instruction time of 8.04 microseconds for IMP-16P system with three clock holds for memory-reference instructions.

Table 1. Average Instructions Executed per Character

rates. Also included are status flags to indicate parity errors, framing errors and overrun errors, control signals for Receiver Data Available (RDA), Transmitter Buffer Empty ($\overline{\text{TBMT}}$), Control Strobe (CS), Receiver Data Enable ($\overline{\text{RDE}}$), Status Word Enable ($\overline{\text{SWE}}$), Transmitter Data Strobe ($\overline{\text{TDS}}$), and Transmitter End of Character (TEOC). These bits comprise the UAR/T status flag register.

SYSTEM IMPLEMENTATION

For breadboarding convenience, the UAR/T was interfaced to an IMP-16P prototyping system. Nevertheless, the principle is the same for any IMP-16 application. Refer to figure 2 for a schematic diagram of the circuit used to interface the IMP-16P with a serial input/output peripheral unit.



(Signal Mnemonics, Refer to IMP-16P System)

Figure 3. UAR/T Timing Diagram

In this application, memory-reference instructions are used rather than peripheral-communication instructions to make use of the clock holds inherent in the IMP-16P prototyping system. This is necessary as the data-hold time on the UAR/T for a load is greater than the 700 nanoseconds available for data transfer with Register In (RIN) and Register Out (ROUT) instructions. Figure 3 is a timing diagram for the UAR/T circuit.

Peripheral decoding is simple and straightforward. Four address lines are used to select the UAR/T. Address lines ADX14* and ADX15 produce the absolute device address; ADX00 and ADX01 select the order code. This produces four order codes each for loads and stores: 8000_{16}, 8001_{16}, 8002_{16}, and 8003_{16}.

A flowchart of the Interrupt Service Routine for receiving and transmitting serial input/output data is shown in figure 4.

In operation, the UAR/T initially is sent a master reset signal to clear all registers and to reset status flags. Next, an 8-bit word is sent to set the baud rate, parity selections, and word-length selection. In this application, the send and receive baud rates are the same. To enter the transmit mode of operation, the first character is sent to the UAR/T via a Store Instruction; then, the transmit enable flag (user flag 11) is set to enable interrupts by the UAR/T when the Holding Register is empty. This latter event indicates that the UAR/T is ready to receive another character for transmission. Address- and character-count pointers are updated in preparation for the next character of the block and the service routine is exited, to be reentered when the next TBMT interrupt occurs.

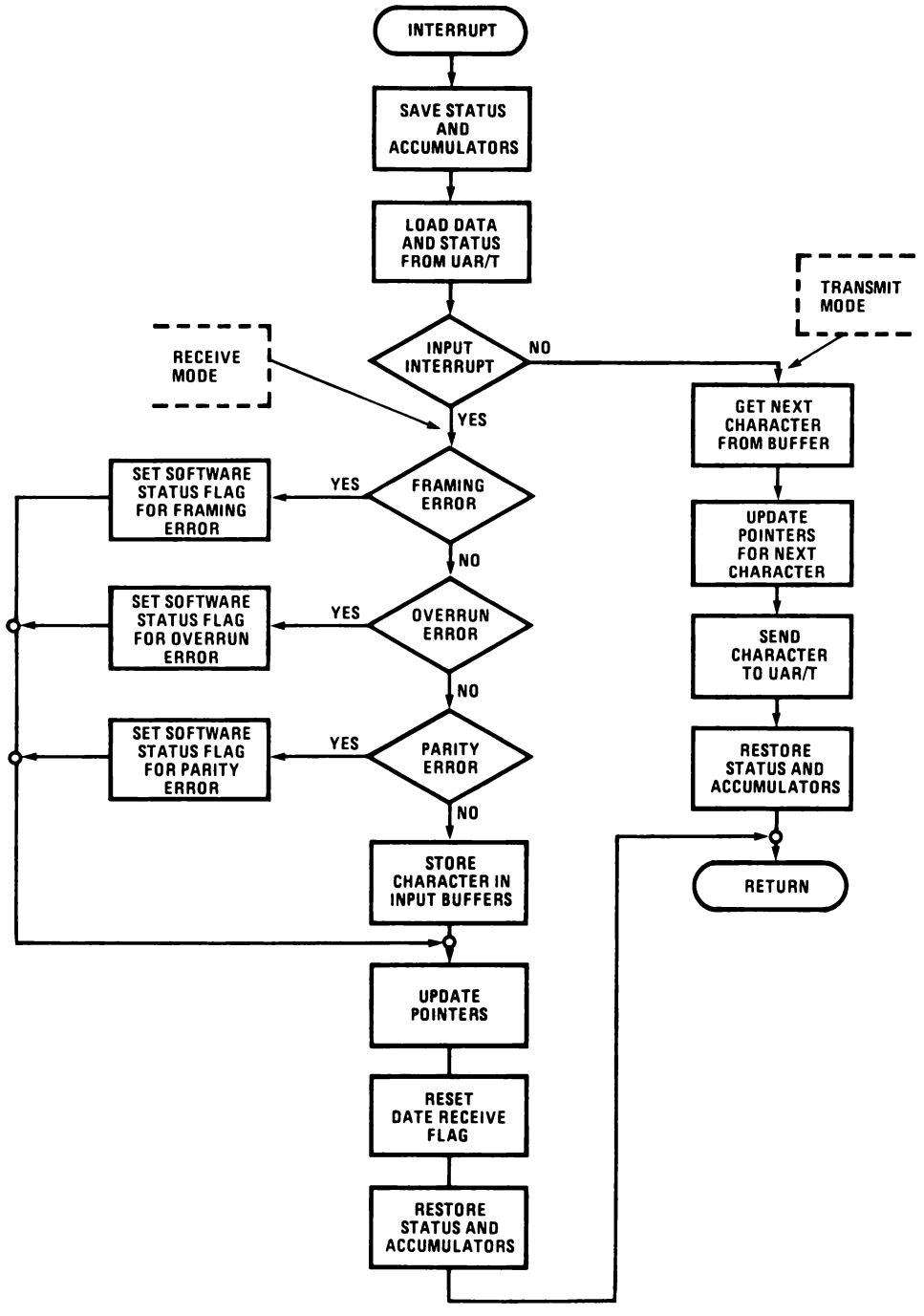


Figure 4. Interrupt Service Routine

In the receive mode, an interrupt is received when a character has been placed in the Receiver Register and the CPU then executes a Load Instruction from the UAR/T; thus, the character and the contents of the UAR/T status register are brought into the specified accumulator. The service routine tests the status of the UAR/T to determine if the interrupt is a receive or transmit interrupt. If transmit mode is indicated, the next character is sent to the UAR/T. If in the receive mode, the error flags are tested. If errors exist, the error routines are executed. If no errors exist, the character, now in the lower 8 bits of the accumulator, is stored in an input table. The Interrupt Service Routine has the task of updating the pointers and the contents of the Output Data Table (ODATA) and Input Data Table (IDATA). Processing of input data is left to the main program. An example of an interrupt service routine used with this application is given in appendix A.

There is no enable on receiver interrupts, as the occurrence of incoming data is unpredictable. Thus, this interrupt is always enabled.

On the transmitter output (TSO) and receiver input (RSI), level translators (LM1488, LM1489) are used to convert the TTL levels of the UAR/T to RS 232C levels.

Any baud rate from 50 to 9600 is available simply by changing the control character sent to the baud-rate-generation circuit. This consists of a crystal-controlled oscillator, latch and programmable frequency divider (MM5307). The oscillator runs at a frequency of 921.60 kilohertz and the divider produces a clock that is 16 times the desired baud rate.

The timing diagram, figure 2, shows the signal relationships for sending and receiving characters. Note the extend time of T4 (T4 + 1, T4 + 2, T4 + 3) to allow additional access time for a memory-reference instruction.

The receiving devices for this project were a Texas Instruments ASR Silent 700 terminal operated at 300 baud and a Lear-Siegler ADM-1 and ADM-2 CRT terminal operated at 1200 and 9600 baud, respectively. The test program continuously transmitted a block of ASCII characters while receiving a block from the terminal. When a block length was received, it was transmitted back to the inputting terminal.

CONCLUSIONS

As can be seen from the test and the graph, figure 1 and table 1, the use of a device such as a UAR/T can increase greatly communication throughput with a small increase in component count. System integrity is similarly improved. This type of serial input/output can be applied easily to an end-user application designed around the IMP-16 chip set, or it can be used at the card or prototyping system level for communication with any asynchronous peripheral.

Appendix A

```
 1                          TITLE  UART, '-SERIAL I/O VIA UAR/T'
 2
 3
 4       0000               .ASECT
 5
 6       0001               .=1
 7
 8       0000     AC0       =        0
 9       0001     AC1       =        1
10       0002     AC2       =        2
11       0003     AC3       =        3
12       0003     BIT0      =        3                ; BIT 0 OF AC0 = 1
13       0004     BIT1      =        4                ; BIT 1 OF AC0 = 1
14       000B     BIT15     =        11               ; BIT 15 OF AC0 = 1 ( AC0<0 )
15       0003     XMITEN    =        3                ; USER FLAG 11 = XMIT ENABLE
16       0001     INTEN     =        1                ; MASTER INTERRUPT ENABLE
17       0001     ZERO      =        1                ; AC0 = 0
18
19                ; INTERRUPT SERVICE ROUTINE
20
21                ; INTERRUPT CONDITIONS:
22
23                ;     BIT 15 TRUE = RECEIVE
24                ;     BIT 14 TRUE = SEND
25
26                ;STATUS CONDITIONS:
27
28                ;     BIT 0 = FRAMING ERROR
29                ;     BIT 1 = OVERRUN ERROR
30                ;     BIT 2 = PARITY ERROR
31
32
33 0001 A030 A  INTRPT: ST       AC0,SAVE         ; SAVE ACCUMULATORS
34 0002 0431 0          ST       AC1,SAVE+1
35 0003 A832 A          ST       AC2,SAVE+2
36 0004 AC33 A          ST       AC3,SAVE+3
37 0005 0080 A          PUSHF                     ; SAVE CPU STATUS FLAGS
38 0006 4400 0          PULL     AC0
```

```
 39 0007 A034 A           ST      AC0,FLAGS
 40 0008 903D A           LD      AC0,@RECV       ; GET CHARACTER AND STATUS
 41 0009 1D09 A           BOC     BIT15,DIN       ; CHECK FOR RECEIVE OR SEND
 42
 43
 44 000A 7838 A  DOUT:    ISZ     OBPTR           ; INCREMENT OUTPUT POINTER
 45 000B 7C2F A           DSZ     OBCTR           ; DECREMENT WORD COUNT
 46 000C 200E A           JMP      +2
 47 000D 2011 A           JMP     TDONE           ; TRANSMITTION COMPLETE
 48 000E 9038 A           LD      AC0,@OBPTR      ; GET NEXT CHARACTER
 49 000F B03A A           ST      AC0,@SEND       ; SEND CHARACTER TO UAR/T
 50 0010 201D A           JMP     RETURN
 51 0011 4700 A  TDONE:   PULL    AC3             ; HOUSEKEEP STACK
 52 0012 2442 A           JMP     @XDONE          ; BLOCK COMPLETE
 53
 54
 55 0013 1317 A  DIN:     BOC     BIT0,FE         ; CHECK FOR FRAMING ERROR
 56 0014 1410 A           BOC     BIT1,OE         ; CHECK FOR OVERRUN ERROR
 57 0015 58FD A           ROR     AC0,3           ; ROTATE BIT 2 TO BIT 15
 58 0016 1B11 A           BOC     BIT15,PE        ; CHECK FOR PARITY ERROR
 59 0017 6043 A           AND     AC0,MASK        ; MASK UPPER 8 BITS
 60 0018 B039 A           ST      AC0,@INBUF      ; STORE CHAR. FOR PROCESSING
 61 0019 B03C A           ST      AC0,@DRRST      ; RESET DATA RECEIVED FLAG
 62 001A 7839 A           ISZ     INBUF           ; INCREMENT INPUT POINTER
 63 001B 782E A           ISZ     INBPTR          ; INCREMENT CHARACTER COUNT
 64 001C 201D A           JMP     RETURN          ; RETURN FROM INTERRUPT
 65
 66
 67 001D 8034 A  RETURN:  LD      AC0,FLAGS       ; RESTORE CPU STATUS FLAGS
 68 001E 4000 A           PUSH    AC0
 69 001F 0280 A           PULLF
 70 0020 8030 A           LD      AC0,SAVE        ; RESTORE ACCUMULATORS
 71 0021 8431 A           LD      AC1,SAVE+1
 72 0022 8832 A           LD      AC2,SAVE+2
 73 0023 8C33 A           LD      AC3,SAVE+3
 74 0024 0100 A           RTI                     ; RETURN FROM INTERRUPT
 75
 76
 77               ; SERVICE ROUTINES FOR STATUS ERRORS DETECTED BY UAR/T.
 78               ;   SOFTWARE FLAGS WILL BE SET IN MEMORY FOR MAIN PROGRAM
 79               ;   TO EXAMINE AND ACT UPON.
 80
 81
 82 0025 4C01 A  OE:      LI      AC0,1
 83 0026 A035 A           ST      AC0,OEFLAG      ; SET SOFTWARE FLAG FOR
 84 0027 201D A           JMP     RETURN          ;   OVERRUN ERROR.
 85
 86
 87 0028 4C01 A  PE:      LI      AC0,1
 88 0029 A036 A           ST      AC0,PEFLAG      ; SET SOFTWARE FLAG FOR
 89 002A 201D A           JMP     RETURN          ;   PARITY ERROR.
 90
 91
 92 002B 4C01 A  FE:      LI      AC0,1
 93 002C A037 A           ST      AC0,FEFLAG      ; SET SOFTWARE FLAG FOR
 94 002D 201D A           JMP     RETURN          ;   FRAMING ERROR.
 95
 96
 97
 98
 99               ; TEMPORARY DATA AND CONSTANTS
100
101
102      002F    INBPTR: .=.+1                   ; INPUT CHARACTER COUNTER
103      0030    OBCTR:  .=.+1                   ; OUTPUT CHARACTER COUNTER
104      0034    SAVE:   .=.+4                   ; TEMPORARY LOCATION FOR ACCS
105      0035    FLAGS:  .=.+1                   ; TEMPORARY LOCATION FOR FLAG
106      0036    OEFLAG: .=.+1                   ; FLAG FOR OVERRUN ERROR
107      0037    PEFLAG: .=.+1                   ; FLAG FOR PARITY ERROR
108      0038    FEFLAG: .=.+1                   ; FLAG FOR FRAMING ERROR
109 0038 0200 A  OBPTR:  .WORD   OUTBUF          ; OUTPUT BUFFER POINTER
```

National Semiconductor

AUGUST 1976
Pub. No. 420305226-001A

INS8080A 8-Bit N-Channel Microprocessor

general description

The INS8080A is an 8-bit microprocessor housed in a standard, 40-pin dual-in-line package. The chip, which is fabricated using N-channel silicon gate MOS technology, functions as the central processing unit (CPU) in National Semiconductor's N8080 microcomputer family.

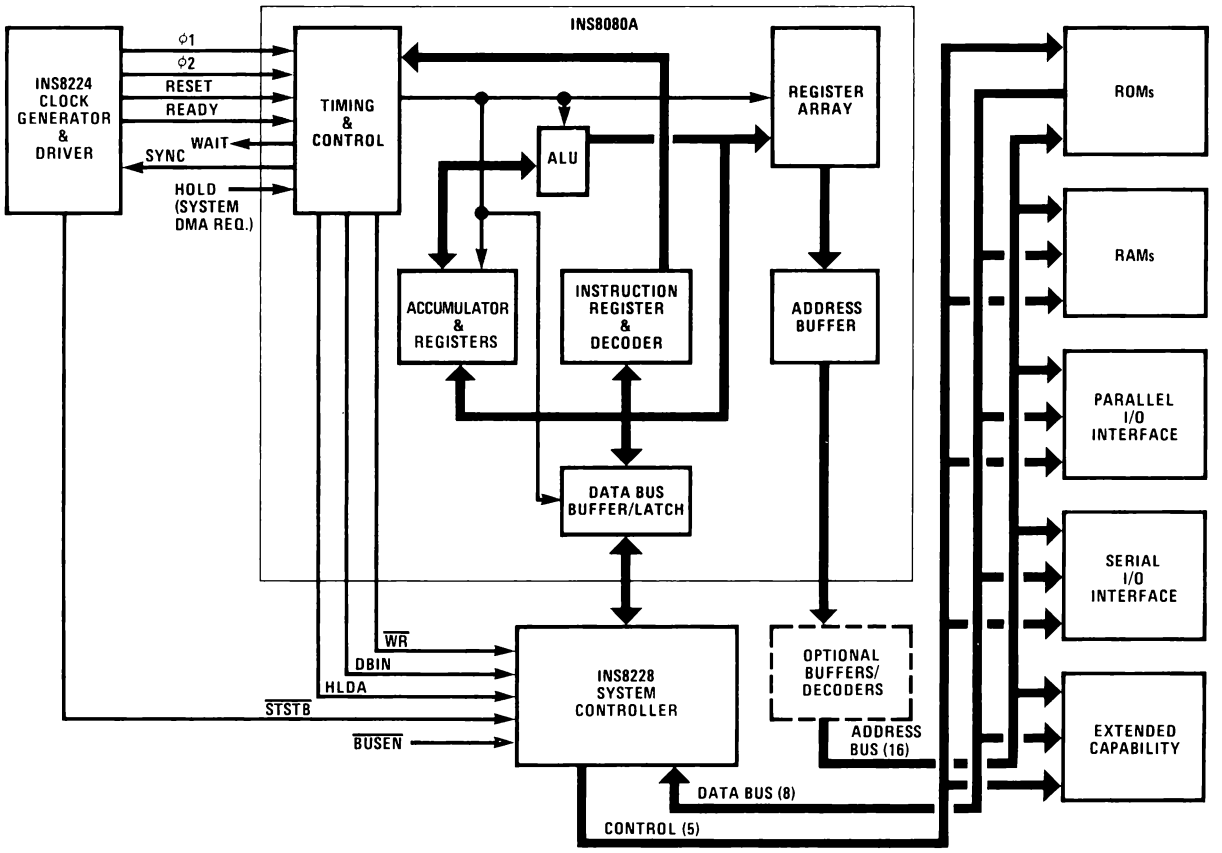
The INS8080A has a 16-bit address bus that is capable of addressing up to 65k bytes of memory and up to 256 input and 256 output devices. Data is routed to and from the INS8080A on a separate bidirectional 8-bit bus. This data bus is also TRI-STATE®, making direct memory addressing (DMA) and multiprocessing applications possible. The INS8080A directly provides signals to control the interface to memory and I/O ports. All buses, including control, are TTL compatible.

An asynchronous interrupt capability is included in the INS8080A to allow external signals to change the instruction sequence. The interrupting device may vector the program to a particular service routine location (or some other direct function) by specifying an interrupt instruction to be executed.

features

- 74 Instructions – Variable Length
- General Purpose Registers – Six plus an Accumulator
- Direct Addressing up to 65k Bytes
- Variable Length Stack Accessed by 16-bit Stack Pointer
- Addresses 256 Input and 256 Output Ports
- Provisions for Vectored Interrupts
- TRI-STATE® Bus for DMA and Multiprocessing Capability
- TRI-STATE TTL Drive Capabilities for Address and Data Buses
- Decimal Arithmetic Capability
- Multiple Addressing Modes
  - Direct
  - Register
  - Register Indirect
  - Immediate
- Direct Plug-in Replacement for Intel 8080A

N8080A microcomputer family block diagram



absolute maximum ratings

Temperature Under Bias . . . . . 0°C to +70°C
Storage Temperature . . . . . -65°C to +150°C
All Input or Output Voltages
with Respect to V_{BB} . . . . . -0.3V to +20V
V_{CC}, V_{DD} and V_{SS} with Respect to V_{BB}. . -0.3V to +20V
Power Dissipation 1.5W

Note: Maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under dc electrical characteristics.

dc electrical characteristics

T_A = 0°C to +70°C, V_{DD} = +12V ± 5%, V_{CC} = +5V ± 5%, V_{BB} = -5V ± 5%, V_{SS} = 0V, unless otherwise noted.

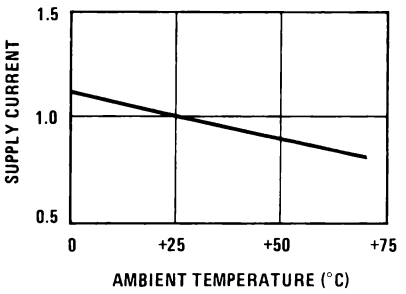
Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	V_{SS}-1		V_{SS}+0.8	V	
V_{IHC}	Clock Input High Voltage	9.0		V_{DD}+1	V	
V_{IL}	Input Low Voltage	V_{SS}-1		V_{SS}+0.8	V	
V_{IH}	Input High Voltage	3.3		V_{CC}+1	V	
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 1.9mA on all outputs,
V_{OH}	Output High Voltage	3.7			V	I_{OH} = 150μA.
$I_{DD\,(AV)}$	Avg. Power Supply Current (V_{DD})		40	70	mA	Operation
$I_{CC\,(AV)}$	Avg. Power Supply Current (V_{CC})		60	80	mA	t_{CY} = 0.48μs
$I_{BB\,(AV)}$	Avg. Power Supply Current (V_{BB})		0.01	1	mA	
I_{IL}	Input Leakage			±10	μA	$V_{SS} \leq V_{IN} \leq V_{CC}$
I_{CL}	Clock Leakage			±10	μA	$V_{SS} \leq V_{CLOCK} \leq V_{DD}$
I_{DL}[2]	Data Bus Leakage in Input Mode			-100	μA	$V_{SS} \leq V_{IN} \leq V_{SS}$ + 0.8V
				-2.0	mA	V_{SS} + 0.8V $\leq V_{IN} \leq V_{CC}$
I_{FL}	Address and Data Bus Leakage			+10	μA	$V_{ADDR/DATA} = V_{CC}$
	During HOLD			-100	μA	$V_{ADDR/DATA} = V_{SS}$ + 0.45V

capacitance

T_A = 25°C, V_{CC} = V_{DD} = V_{SS} = 0V, V_{BB} = -5V

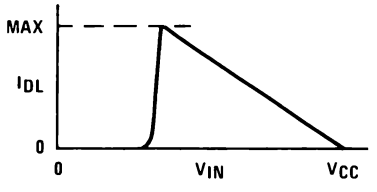
Symbol	Parameter	Typ.	Max.	Unit	Test Condition
C_ϕ	Clock Capacitance	17	25	pF	f_c = 1MHz
C_{IN}	Input Capacitance	6	10	pF	Unmeasured Pins
C_{OUT}	Output Capacitance	10	20	pF	Returned to V_{SS}

TYPICAL SUPPLY CURRENT VS. TEMPERATURE, NORMALIZED(3)



Notes:
1. The RESET signal must be active for a minimum of 3 clock cycles.
2. When DBIN is high and $V_{IN} > V_{IH}$ an internal active pullup will be switched onto the Data Bus.
3. $\Delta I_{supply} / \Delta T_A$ = -0.45%/°C.

DATA BUS CHARACTERISTIC DURING DBIN

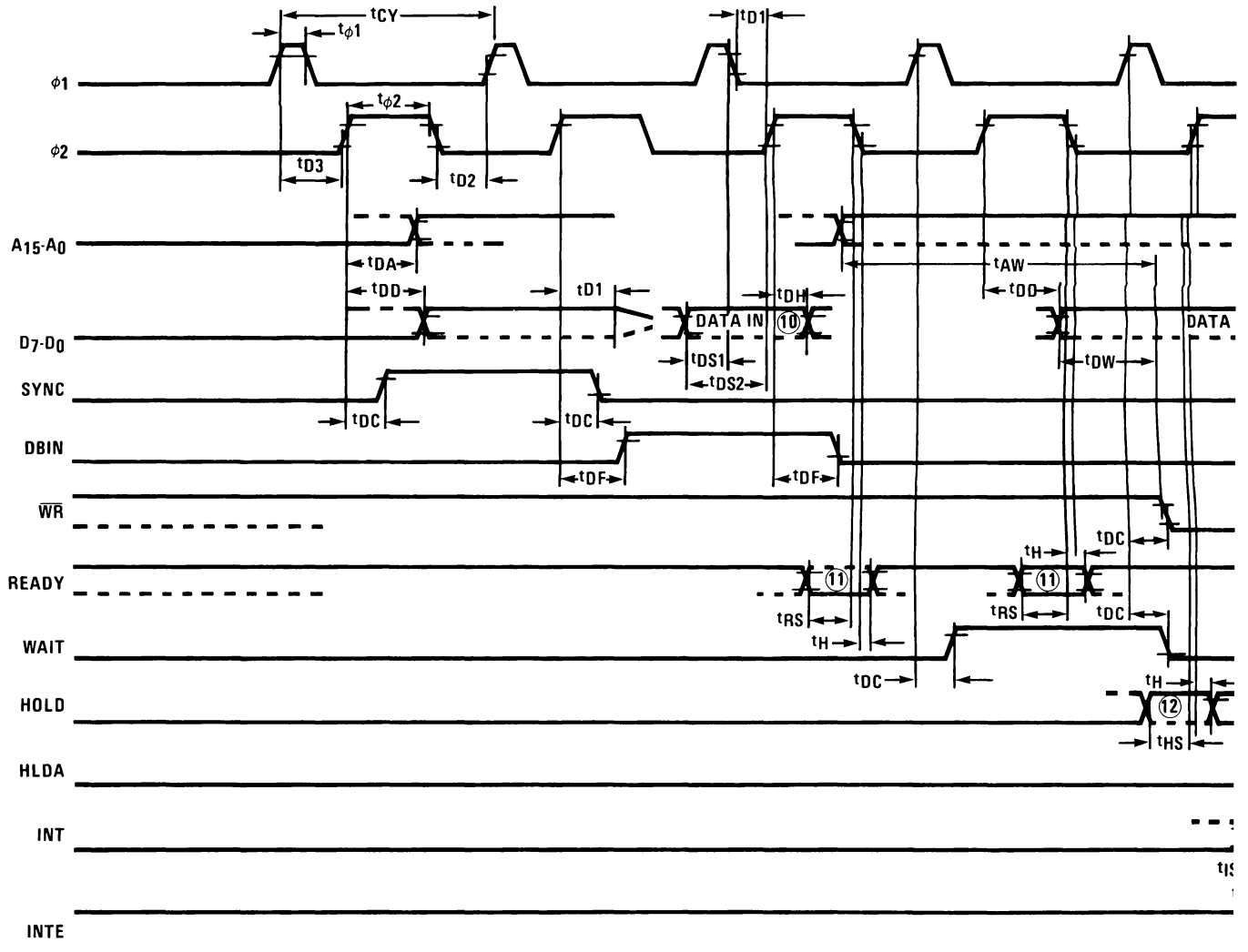


ac electrical characteristics

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{CY} [3]	Clock Period	0.48	2.0	μs	
t_r, t_f	Clock Rise and Fall Time	0	50	ns	
$t_{\phi 1}$	ϕ_1 Pulse Width	60		ns	
$t_{\phi 2}$	ϕ_2 Pulse Width	220		ns	
t_{D1}	Delay ϕ_1 to ϕ_2	0		ns	
t_{D2}	Delay ϕ_2 to ϕ_1	70		ns	
t_{D3}	Delay ϕ_1 to ϕ_2 Leading Edges	80		ns	
t_{DA} [2]	Address Output Delay from ϕ_2		200	ns	C_L = 100pF
t_{DD} [2]	Data Output Delay from ϕ_2		220	ns	
t_{DC} [2]	Signal Output Delay from ϕ_1 or ϕ_2 (SYNC, $\overline{WR}$, WAIT, HLDA)		120	ns	C_L = 50pF
t_{DF} [2]	DBIN Delay from ϕ_2	25	140	ns	
t_{DI} [1]	Delay for Input Bus to Enter Input Mode		t_{DF}	ns	
t_{DS1}	Data Setup Time During ϕ_1 and DBIN	30		ns	

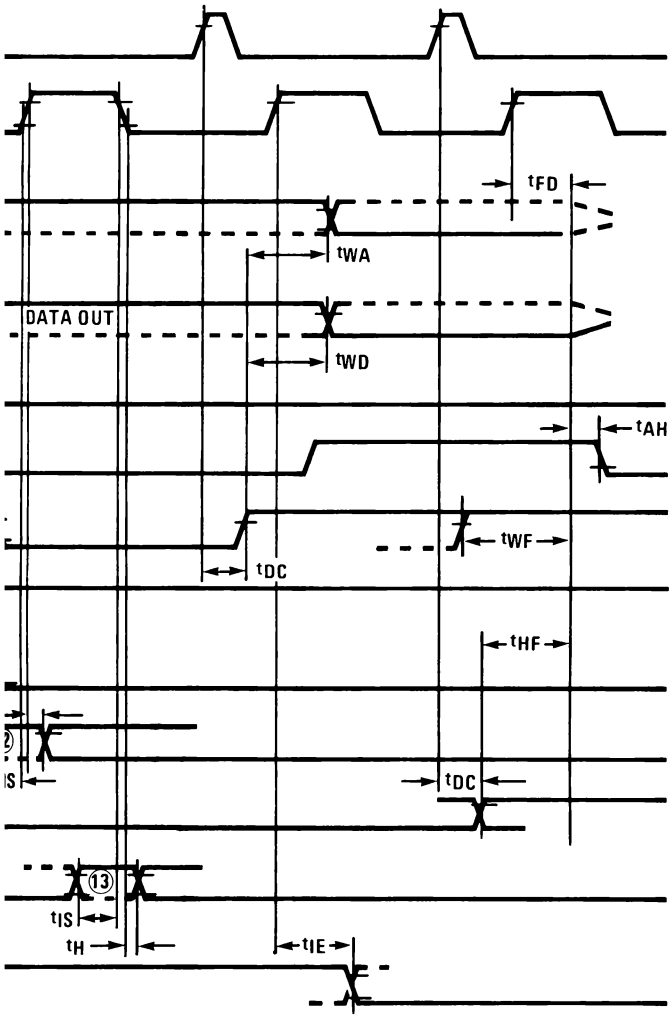
timing waveforms [14]

Note: Timing measurements are made at the following reference voltages: CLOCK '1' = 8.0V, '0' = 1.0V; INPUTS '1' = 3.3V, '0' = 0.8V; OUTPUTS '1' = 2.0V, '0' = 0.8V.



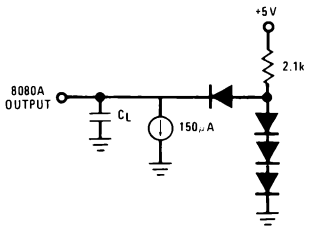
ac electrical characteristics (cont'd.)

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{DS2}	Data Setup Time to ϕ_2 During DBIN	150		ns	
t_{DH} [1]	Data Hold Time from ϕ_2 During DBIN	1		ns	
t_{IE} [2]	INTE Output Delay from ϕ_2		200	ns	C_L = 50pF
t_{RS}	READY Setup Time During ϕ_2	120		ns	
t_{HS}	HOLD Setup Time to ϕ_2	140		ns	
t_{IS}	INT Setup Time During ϕ_2 (During ϕ_1 in Halt Mode)	120		ns	
t_H	Hold Time from ϕ_2 (READY, INT, HOLD)	0		ns	
t_{FD}	Delay to Float During Hold (Address and Data Bus)		120	ns	
t_{AW} [2]	Address Stable Prior to $\overline{WR}$	5		ns	C_L = 100pF : Address, Data C_L = 50pF: $\overline{WR}$, HLDA, DBIN
t_{DW} [2]	Output Data Stable Prior to $\overline{WR}$	6		ns	
t_{WD} [2]	Output Data Stable from $\overline{WR}$	7		ns	
t_{WA} [2]	Address Stable from $\overline{WR}$	7		ns	
t_{HF} [2]	HLDA to Float Delay	8		ns	
t_{WF} [2]	$\overline{WR}$ to Float Delay	9		ns	
t_{AH} [2]	Address Hold Time After DBIN During HLDA	-20		ns	

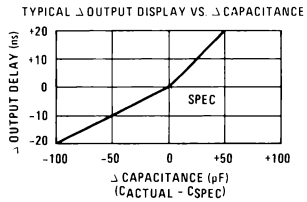


Notes:

1. Data input should be enabled with DBIN status. No bus conflict can then occur and data hold time is assured. t_{DH} = 50ns or t_{DF}, whichever is less.
2. Typical load circuit:

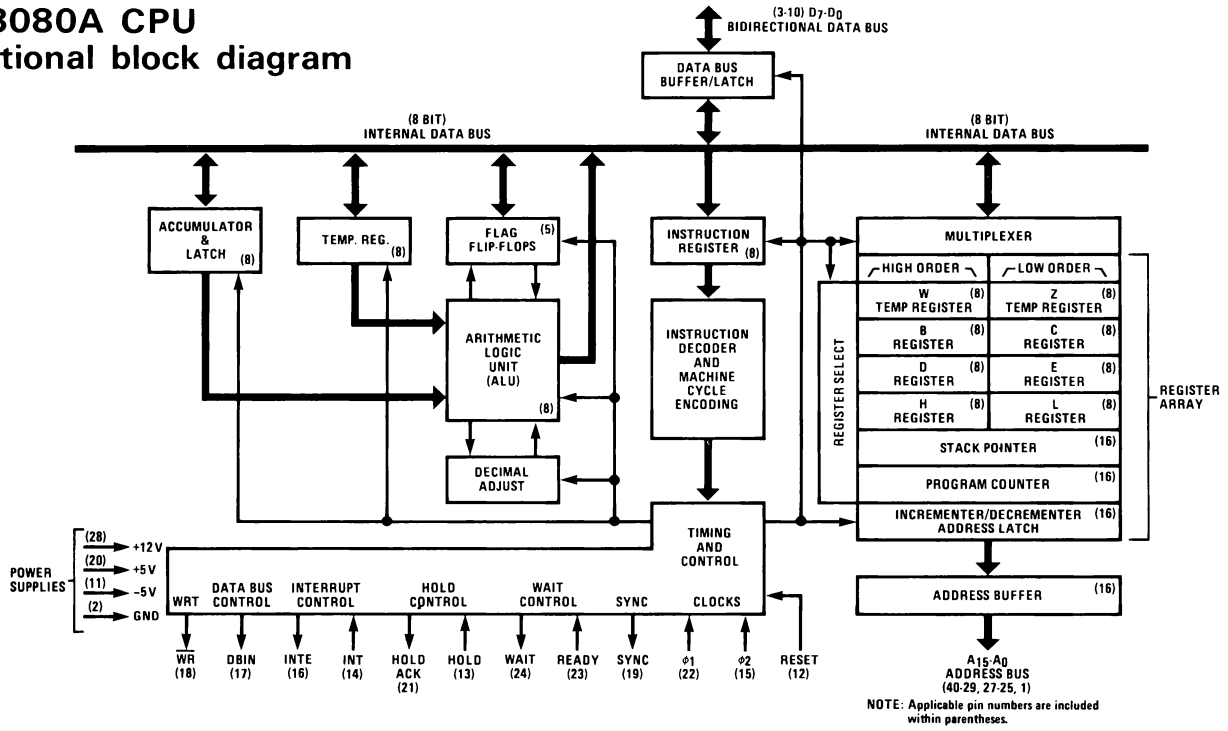


3. $t_{CY} = t_{D3} + t_{r\phi2} + t_{\phi2} + t_{D2} + t_{f\phi2} + t_{r\phi1} \geq 480$ns.



4. The following are relevant when interfacing the INS8080A to devices having V_{IH} = 3.3V:
   a) Maximum output rise time from 0.8V to 3.3V = 100ns @ C_L = SPEC.
   b) Output Delay when measured to 3.0V = SPEC + 60ns @ C_L = SPEC.
   c) If $C_L \neq$ SPEC, add 0.6ns/pF if $C_L > C_{SPEC}$, subtract 0.3ns/pF (from modified delay) if $C_L < C_{SPEC}$.
5. $t_{AW} = 2t_{CY} - t_{D3} - t_{r\phi2} - 140$ns.
6. $t_{DW} = t_{CY} - t_{D3} - t_{r\phi2} - 170$ns.
7. If not HLDA, $t_{WD} = t_{WA} = t_{D3} + t_{r\phi2} + 10$ns. If HLDA, $t_{WD} = t_{WA} = t_{WF}$.
8. $t_{HF} = t_{D3} + t_{r\phi2} - 50$ns.
9. $t_{WF} = t_{D3} + t_{r\phi2} - 10$ns.
10. Data in must be stable for this period during DBIN · T_3. Both t_{DS1} and t_{DS2} must be satisfied.
11. Ready signal must be stable for this period during T_2 or T_W. (Must be externally synchronized.)
12. Hold signal must be stable for this period during T_2 or T_W when entering hold mode, and during T_3, T_4, T_5, and T_{WH} when in hold mode. (External synchronization is not required.)
13. Interrupt signal must be stable during this period of the last clock cycle of any instruction in order to be recognized on the following instruction. (External synchronization is not required.)
14. This timing diagram shows timing relationships only; it does not represent any specific machine cycle.

INS8080A CPU functional block diagram



NOTE: Applicable pin numbers are included within parentheses.

INS8080A functional pin definition

The following describes the function of all of the INS8080A input/output pins. Some of these descriptions reference internal timing periods.

INPUT SIGNALS

Ready: When high (logic 1), indicates that valid memory or input data are available to the CPU on the INS8080A data bus. The READY signal is used to synchronize the CPU with slower memory or input/output devices. If the INS8080A does not receive a high READY input after sending out an address to memory or an input/output device, the INS8080A enters a WAIT mode for as long as the READY input remains low (logic 0). The CPU may also be single stepped by the use of the READY signal.

Hold: When high, requests that the CPU enter the HOLD mode. When the CPU is in the HOLD mode, the CPU address and data buses both will be in the high-impedance state. The HOLD mode allows an external device to gain control of the INS8080A address and data buses immediately following the completion of the current machine cycle by the CPU. The CPU acknowledges the HOLD mode via the HOLD ACKNOWLEDGE (HLDA) output line. The HOLD request is recognized under the following conditions:

- The CPU is in the HALT mode.
- The READY signal is active and the CPU is in the t_2 or t_W microcycle.

Interrupt (INT) Request: When high, the CPU recognizes an interrupt request on this line after completing the current instruction or while in the HALT mode. An interrupt request is not honored if the CPU is in the HOLD mode (HLDA = logic 1) or the Interrupt Enable Flip-flop is reset (INTE = logic 0).

Reset: When activated (high) for a minimum of three clock periods, the content of the Program Counter is cleared and the Interrupt Enable and Hold Acknowledge Flip-flops are reset. Following a RESET, program execution starts at memory location 0. It should be noted that the status flags, accumulator, stack pointer, and registers are not cleared during the RESET sequence.

ϕ_1 and ϕ_2 Clocks: Two non-TTL compatible clock phases which provide nonoverlapping timing references for internal storage elements and logic circuits of the CPU.

+12 Volts: V_{DD} Supply.

+5 Volts: V_{CC} Supply.

–5 Volts: V_{BB} Supply.

Ground: V_{SS} (0 volt) reference.

OUTPUT SIGNALS

Synchronizing (SYNC) Signal: When activated (high), the beginning of a new machine cycle is indicated and the status word is outputted on the Data Bus.

Address (A_{15} – A_0) Bus: This bus comprises sixteen TRI-STATE output lines. The bus provides the address to memory (up to 65k bytes) or denotes the input/output device number for up to 256 input and 256 output peripherals.

Wait: When high, acknowledges that the CPU is in the WAIT mode.

Write ($\overline{WR}$): When low, the data on the data bus are stable for WRITE memory or output operation.

Hold Acknowledge (HLDA): Goes high in response to a logic 1 on the HOLD line and indicates that the data and address bus will go to the high-impedance state. The HLDA begins at one of the following times:

- The t_3 microcycle of a READ memory input operation.
- The clock period following the t_3 microcycle of a WRITE memory output operation.

In both cases, the HLDA signal starts after the rising edge of the ϕ_1 clock, and high impedance occurs after the rising edge of the ϕ_2 clock.

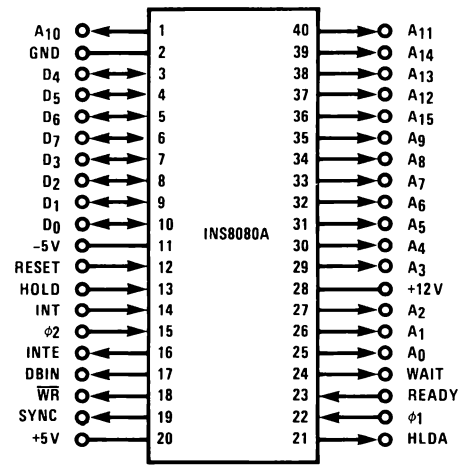
Interrupt Enable (INTE): Indicates the content of the internal Interrupt Enable Flip-flop. The Enable and Disable Interrupt (EI and DI) Instructions cause the Interrupt Enable Flip-flop to be set and reset, respectively. When the flip-flop is reset (INTE = logic 0), it inhibits interrupts from being accepted by the CPU. In addition, the Interrupt Enable Flip-flop is automatically reset (thereby disabling further interrupts) at the t_1 microcycle of the instruction fetch cycle, when an interrupt is accepted; it is also reset by the RESET Signal.

Data Bus In (DBIN): When high, indicates to external circuits that the data bus is in the input mode. The DBIN Signal should be used to gate data from memory or an I/O device onto the Data Bus.

INPUT/OUTPUT SIGNALS

Data ($D_7 - D_0$) Bus: This bus comprises eight TRI-STATE input/output lines. The bus provides bidirectional communication between the CPU, memory, and input/output devices for instructions and data transfers. A status word (which describes the current machine cycle) is also outputted on the data bus during the first microcycle of each machine cycle (SYNC = logic 1).

pin configuration



8080A status

Instructions for the 8080A require from one to five machine cycles for complete execution. The 8080A sends out 8 bits of status information on the data bus at the beginning of each machine cycle (during SYNC time). The following table defines the status information.

Status Information Definition

Symbols	Data Bus Bit	Definition
INTA*	D_0	Acknowledge signal for INTERRUPT request. Signal should be used to gate a restart instruction onto the data bus when DBIN is active.
$\overline{WO}$	D_1	Indicates that the operation in the current machine cycle will be a WRITE memory or OUTPUT function (WO = 0). Otherwise, a READ memory or INPUT operation will be executed.
STACK	D_2	Indicates that the address bus holds the pushdown stack address from the Stack Pointer.
HLTA	D_3	Acknowledge signal for HALT Instruction.
OUT	D_4	Indicates that the address bus contains the address of an output device and the data bus will contain the output data when WR is active.
M_1	D_5	Provides a signal to indicate that the CPU is in the fetch cycle for the first byte of an instuction.
INP*	D_6	Indicates that the address bus contains the address of an input device and the input data should be placed on the data bus when DBIN is active.
MEMR*	D_7	Designates that the data bus will be used for memory read data.

*These three status bits can be used to control the flow of data onto the INS8080A data bus.

Status Word Chart

Machine Cycle	Type	Data Bus Bit							
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Instruction Fetch	1	1	0	1	0	0	0	1	0
Memory Read	2	1	0	0	0	0	0	1	0
Memory Write	3	0	0	0	0	0	0	0	0
Stack Read	4	1	0	0	0	0	1	1	0
Stack Write	5	0	0	0	0	0	1	0	0
Input Read	6	0	1	0	0	0	0	1	0
Output Write	7	0	0	0	1	0	0	0	0
Interrupt Acknowledge	8	0	0	1	0	0	0	1	1
Halt Acknowledge	9	1	0	0	0	1	0	1	0
Interrupt Acknowledge While Halt	10	0	0	1	0	1	0	1	1

instruction set

Mnemonic	Description	Operation	Op Code D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	No. of Bytes	No. of Machine (M) Cycles	No. of μcycles (T)	Condition Flags S	Z	AC	P	CY
DATA TRANSFER GROUP																		
LDA	Load Accumulator Direct	(A) ← ((byte 3)(byte 2))	0	0	1	1	1	0	1	0	3	4	13					
LDAX B	Load Accumulator Indirect	(A) ← ((B))	0	0	0	0	1	0	1	0	1	2	7					
LDAX D	Load Accumulator Indirect	(A) ← ((D))	0	0	0	1	1	0	1	0	1	2	7					
LHLD	Load H and L Direct	(L) ← ((byte 3)(byte 2)) (H) ← ((byte 3)(byte 2) + 1)	0	0	1	0	1	0	1	0	3	5	16					
LXI B	Load Immediate, Registers B and C	(B) ← (byte 3) (C) ← (byte 2)	0	0	0	0	0	0	0	1	3	3	10					
LXI D	Load Immediate, Registers D and E	(D) ← (byte 3) (E) ← (byte 2)	0	0	0	1	0	0	0	1	3	3	10					
LXI H	Load Immediate, Registers H and L	(H) ← (byte 3) (L) ← (byte 2)	0	0	1	0	0	0	0	1	3	3	10					
LXI SP	Load Immediate, Stack Pointer	(SPH) ← (byte 3) (SPL) ← (byte 2)	0	0	1	1	0	0	0	1	3	3	10	(Flags Not Affected)				
MOV M, r	Move to Memory	((H)(L)) ← (r)	0	1	1	1	0	S	S	S	1	2	7					
MOV r, M	Move from Memory	(r) ← ((H)(L))	0	1	D	D	D	1	1	0	1	2	7					
MOV r1, r2	Move Registers	(r1) ← (r2)	0	1	D	D	D	S	S	S	1	1	5					
MVI M	Move to Memory Immediate	((H)(L)) ← (byte 2)	0	0	1	1	0	1	1	0	2	3	10					
MVI r	Move Immediate	(r) ← (byte 2)	0	0	D	D	D	1	1	0	2	2	7					
SHLD	Store H and L Direct	((byte 3)(byte 2)) ← (L) ((byte 3)(byte 2) + 1) ← (H)	0	0	1	0	0	0	1	0	3	5	16					
STA	Store Accumulator Direct	((byte 3)(byte 2)) ← (A)	0	0	1	1	0	0	1	0	3	4	13					
STAX B	Store Accumulator Indirect	((B)) ← (A)	0	0	0	0	0	0	1	0	1	2	7					
STAX D	Store Accumulator Indirect	((D)) ← (A)	0	0	0	1	0	0	1	0	1	2	7					
XCHG	Exchange H and L with D and E	(H) ↔ (D) (L) ↔ (E)	1	1	1	0	1	0	1	1	1	1	4					
ARITHMETIC GROUP																		
ACI	Add Immediate with Carry	(A) ← (A) + (byte 2) + (CY)	1	1	0	0	1	1	1	0	2	2	7	‡	‡	‡	‡	‡
ADC M	Add Memory with Carry	(A) ← (A) + ((H)(L)) + (CY)	1	0	0	0	1	1	1	0	1	2	7	‡	‡	‡	‡	‡
ADC r	Add Register with Carry	(A) ← (A) + (r) + (CY)	1	0	0	0	1	S	S	S	1	1	4	‡	‡	‡	‡	‡
ADD M	Add Memory	(A) ← (A) + ((H)(L))	1	0	0	0	0	1	1	0	1	2	7	‡	‡	‡	‡	‡
ADD r	Add Register	(A) ← (A) + (r)	1	0	0	0	0	S	S	S	1	1	4	‡	‡	‡	‡	‡
ADI	Add Immediate	(A) ← (A) + (byte 2)	1	1	0	0	0	1	1	0	1	2	7	‡	‡	‡	‡	‡
DAA	Decimal Adjust Accumulator	8-bit number in Accumulator is converted to two 4-bit BCD digits	0	0	1	0	0	1	1	1	1	1	4	‡	‡	‡	‡	‡
DAD B	Add B and C to H and L	(H)(L) ← (H)(L) + (B)(C)	0	0	0	0	1	0	0	1	1	3	10	·	·	·	·	‡
DAD D	Add D and E to H and L	(H)(L) ← (H)(L) + (D)(E)	0	0	0	1	1	0	0	1	1	3	10	·	·	·	·	‡
DAD H	Add H and L to H and L	(H)(L) ← (H)(L) + (H)(L)	0	0	1	0	1	0	0	1	1	3	10	·	·	·	·	‡
DAD SP	Add Stack Pointer to H and L	(H)(L) ← (H)(L) + (SP)	0	0	1	1	1	0	0	1	1	3	10	·	·	·	·	‡
DCR M	Decrement Memory	((H)(L)) ← ((H)(L)) - 1	0	0	1	1	0	1	0	1	1	3	10	‡	‡	‡	‡	·
DCR r	Decrement Register	(r) ← (r) - 1	0	0	D	D	D	1	0	1	1	1	5	‡	‡	‡	‡	·
DCX B	Decrement Registers B and C	(B)(C) ← (B)(C) - 1	0	0	0	0	1	0	1	1	1	1	5	·	·	·	·	·
DCX D	Decrement Registers D and E	(D)(E) ← (D)(E) - 1	0	0	0	1	1	0	1	1	1	1	5	·	·	·	·	·
DCX H	Decrement Registers H and L	(H)(L) ← (H)(L) - 1	0	0	1	0	1	0	1	1	1	1	5	·	·	·	·	·
DCX SP	Decrement Stack Pointer	(SP) ← (SP) - 1	0	0	1	1	1	0	1	1	1	1	5	·	·	·	·	·
INR M	Increment Memory	((H)(L)) ← ((H)(L)) + 1	0	0	1	1	0	1	0	0	1	3	10	‡	‡	‡	‡	·
INR r	Increment Register	(r) ← (r) + 1	0	0	D	D	D	1	0	0	1	1	5	‡	‡	‡	‡	·
INX B	Increment Registers B and C	(B)(C) ← (B)(C) + 1	0	0	0	0	0	0	1	1	1	1	5	·	·	·	·	·
INX D	Increment Registers D and E	(D)(E) ← (D)(E) + 1	0	0	0	1	0	0	1	1	1	1	5	·	·	·	·	·
INX H	Increment Registers H and L	(H)(L) ← (H)(L) + 1	0	0	1	0	0	0	1	1	1	1	5	·	·	·	·	·
INX SP	Increment Stack Pointer	(SP) ← (SP) + 1	0	0	1	1	0	0	1	1	1	1	5	·	·	·	·	·
SBB M	Subtract Memory with Borrow	(A) ← (A) - ((H)(L)) - (CY)	1	0	0	1	1	1	1	0	1	2	7	‡	‡	‡	‡	‡
SBB r	Subtract Register with Borrow	(A) ← (A) - (r) - (CY)	1	0	0	1	1	S	S	S	1	1	4	‡	‡	‡	‡	‡
SBI	Subtract Immediate with Borrow	(A) ← (A) - (byte 2) - (CY)	1	1	0	1	1	1	1	0	2	2	7	‡	‡	‡	‡	‡
SUB M	Subtract Memory	(A) ← (A) - ((H)(L))	1	0	0	1	0	1	1	0	1	2	7	‡	‡	‡	‡	‡
SUB r	Subtract Register	(A) ← (A) - (r)	1	0	0	1	0	S	S	S	1	1	4	‡	‡	‡	‡	‡
SUI	Subtract Immediate	(A) ← (A) - (byte 2)	1	1	0	1	0	1	1	0	2	2	7	‡	‡	‡	‡	‡
LOGICAL GROUP																		
ANA M	AND Memory	(A) ← (A) ∧ ((H)(L))	1	0	1	0	0	1	1	0	1	2	7	‡	‡	·	‡	0
ANA r	AND Register	(A) ← (A) ∧ (r)	1	0	1	0	0	S	S	S	1	1	4	‡	‡	·	‡	0
ANI	AND Immediate	(A) ← (A) ∧ (byte 2)	1	1	1	0	0	1	1	0	2	2	7	‡	‡	·	‡	0
CMA	Complement Accumulator	(A) ← ($\overline{A}$)	0	0	1	0	1	1	1	1	1	1	4	·	·	·	·	·
CMC	Complement Carry	(CY) ← ($\overline{CY}$)	0	0	1	1	1	1	1	1	1	1	4	·	·	·	·	‡
CMP M	Compare Memory	(A) — ((H)(L))	1	0	1	1	1	1	1	0	1	2	7	‡	‡[a]	‡	‡	‡[a]
CMP r	Compare Register	(A) — (r)	1	0	1	1	1	S	S	S	1	1	4	‡	‡[b]	‡	‡	‡[b]
CPI	Compare Immediate	(A) — (byte 2)	1	1	1	1	1	1	1	0	2	2	7	‡	‡[c]	‡	‡	‡[c]
ORA M	OR Memory	(A) ← (A) ∨ ((H)(L))	1	0	1	1	0	1	1	0	1	2	7	‡	‡	0	‡	0
ORA r	OR Register	(A) ← (A) ∨ (r)	1	0	1	1	0	S	S	S	1	1	4	‡	‡	0	‡	0
ORI	OR Immediate	(A) ← (A) ∨ (byte 2)	1	1	1	1	0	1	1	0	2	2	7	‡	‡	0	‡	0
RAL	Rotate Left through Carry	$(A_{n+1}) \leftarrow (A_n)$; (CY) ← (A_7) (A_0) ← (CY)	0	0	0	1	0	1	1	1	1	1	4	·	·	·	·	‡
RAR	Rotate Right through Carry	$(A_n) \leftarrow (A_{n+1})$; (CY) ← (A_0) (A_7) ← (CY)	0	0	0	1	1	1	1	1	1	1	4	·	·	·	·	‡
RLC	Rotate Left	$(A_{n+1}) \leftarrow (A_n)$; $(A_0) \leftarrow (A_7)$ (CY) ← (A_7)	0	0	0	0	0	1	1	1	1	1	4	·	·	·	·	‡
RRC	Rotate Right	$(A_n) \leftarrow (A_{n-1})$; $(A_7) \leftarrow (A_0)$ (CY) ← (A_0)	0	0	0	0	1	1	1	1	1	1	4	·	·	·	·	‡
STC	Set Carry	(CY) ← 1	0	0	1	1	0	1	1	1	1	1	4	·	·	·	·	1
XRA M	Exclusive OR Memory	(A) ← (A) ∀ ((H)(L))	1	0	1	0	1	1	1	0	1	2	7	‡	‡	0	‡	0
XRA r	Exclusive OR Register	(A) ← (A) ∀ (r)	1	0	1	0	1	S	S	S	1	1	4	‡	‡	0	‡	0
XRI	Exclusive OR Immediate	(A) ← (A) ∀ (byte 2)	1	1	1	0	1	1	1	0	2	2	7	‡	‡	0	‡	0

Notes:
a. Z = 1 if (A) = (H)(L); CY = 1 if (A) < (H)(L)
b. Z = 1 if (A) = (r); CY = 1 if (A) < (r)
c. Z = 1 if (A) = (byte 2); CY = 1 if (A) < (byte 2)

instruction set (cont'd.)

Mnemonic	Description	Operation	Op Code D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	No. of Bytes	No. of Machine (M) Cycles	No. of μcycles (T)	Condition Flags S	Z	AC	P	CY
BRANCH GROUP																		
CALL	Call Unconditional	((SP) - 1) ← (PCH) ((SP) - 2) ← (PCL) (SP) ← (SP) - 2 (PC) ← (byte 3) (byte 2)	1	1	0	0	1	1	0	1	3	5	17					
CC	Call on Carry	If CY = 1, ((SP) - 1) ← (PCH) ((SP) - 2) ← (PCL) (SP) ← (SP) - 2 (PC) ← (byte 3) (byte 2)	1	1	0	1	1	1	0	0	3	3/5	11/17					
CM	Call on Minus	If S = 1, ((SP) - 1) ← (PCH) ((SP) - 2) ← (PCL) (SP) ← (SP) - 2 (PC) ← (byte 3) (byte 2)	1	1	1	1	1	1	0	0	3	3/5	11/17					
CNC	Call on No Carry	If CY = 0, ((SP) - 1) ← (PCH) ((SP) - 2) ← (PCL) (SP) ← (SP) - 2 (PC) ← (byte 3) (byte 2)	1	1	0	1	0	1	0	0	3	3/5	11/17					
CNZ	Call on Not Zero	If Z = 0, ((SP) - 1) ← (PCH) ((SP) - 2) ← (PCL) (SP) ← (SP) - 2 (PC) ← (byte 3) (byte 2)	1	1	0	0	0	1	0	0	3	3/5	11/17	(Flags Not Affected)				
CP	Call on Positive	If S = 0, ((SP) - 1) ← (PCH) ((SP) - 2) ← (PCL) (SP) ← (SP) - 2 (PC) ← (byte 3) (byte 2)	1	1	1	1	0	1	0	0	3	3/5	11/17					
CPE	Call on Parity Even	If P = 1, ((SP) - 1) ← (PCH) ((SP) - 2) ← (PCL) (SP) ← (SP) - 2 (PC) ← (byte 3) (byte 2)	1	1	1	0	1	1	0	0	3	3/5	11/17					
CPO	Call on Parity Odd	If P = 0, ((SP) - 1) ← (PCH) ((SP) - 2) ← (PCL) (SP) ← (SP) - 2 (PC) ← (byte 3) (byte 2)	1	1	1	0	0	1	0	0	3	3/5	11/17					
CZ	Call on Zero	If Z = 1, ((SP) - 1) ← (PCH) ((SP) - 2) ← (PCL) (SP) ← (SP) - 2 (PC) ← (byte 3) (byte 2)	1	1	0	0	1	1	0	0	3	3/5	11/17					
JC	Jump on Carry	If CY = 1, (PC) ← (byte 3) (byte 2)	1	1	0	1	1	0	1	0	3	3	10					
JM	Jump on Minus	If S = 1, (PC) ← (byte 3) (byte 2)	1	1	1	1	1	0	1	0	3	3	10					
JMP	Jump Unconditional	(PC) ← (byte 3) (byte 2)	1	1	0	0	0	0	1	1	3	3	10					
JNC	Jump on No Carry	If CY = 0, (PC) ← (byte 3) (byte 2)	1	1	0	1	0	0	1	0	3	3	10					
JNZ	Jump on Not Zero	If Z = 0, (PC) ← (byte 3) (byte 2)	1	1	0	0	0	0	1	0	3	3	10					
JP	Jump on Positive	If S = 0, (PC) ← (byte 3) (byte 2)	1	1	1	1	0	0	1	0	3	3	10					
JPE	Jump on Parity Even	If P = 1, (PC) ← (byte 3) (byte 2)	1	1	1	0	1	0	1	0	3	3	10					
JPO	Jump on Parity Odd	If P = 0, (PC) ← (byte 3) (byte 2)	1	1	1	0	0	0	1	0	3	3	10					
JZ	Jump on Zero	If Z = 1, (PC) ← (byte 3) (byte 2)	1	1	0	0	1	0	1	0	3	3	10					
PCHL	H and L to Program Counter	(PCH) ← (H) (PCL) ← (L)	1	1	1	0	1	0	0	1	1	1	5					
RC	Return on Carry	If CY = 1, (PCL) ← ((SP)) (PCH) ← ((SP) + 1) (SP) ← (SP) + 2	1	1	0	1	1	0	0	0	1	1/3	5/11					
RET	Return	(PCL) ← ((SP)); (PCH) ← ((SP) + 1); (SP) ← (SP) + 2;	1	1	0	0	1	0	0	1	1	3	10					
RM	Return on Minus	If S = 1, (PCL) ← ((SP)) (PCH) ← ((SP) + 1) (SP) ← (SP) + 2	1	1	1	1	1	0	0	0	1	1/3	5/11					
RNC	Return on No Carry	If CY = 0, (PCL) ← ((SP)) (PCH) ← ((SP) + 1) (SP) ← (SP) + 2	1	1	0	1	0	0	0	0	1	1/3	5/11					
RNZ	Return on Not Zero	If Z = 0, (PCL) ← ((SP)) (PCH) ← ((SP) + 1) (SP) ← (SP) + 2	1	1	0	0	0	0	0	0	1	1/3	5/11					
RP	Return on Positive	If S = 0, (PCL) ← ((SP)) (PCH) ← ((SP) + 1) (SP) ← (SP) + 2	1	1	1	1	0	0	0	0	1	1/3	5/11					
RPE	Return on Parity Even	If P = 1, (PCL) ← ((SP)) (PCH) ← ((SP) + 1) (SP) ← (SP) + 2	1	1	1	0	1	0	0	0	1	1/3	5/11					
RPO	Return on Parity Odd	If P = 0, (PCL) ← ((SP)) (PCH) ← ((SP) + 1) (SP) ← (SP) + 2	1	1	1	0	0	0	0	0	1	1/3	5/11					

instruction set (cont'd.)

Mnemonic	Description	Operation	Op Code D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	No. of Bytes	No. of Machine (M) Cycles	No. of μcycles (T)	Condition Flags S	Z	AC	P	CY
BRANCH GROUP (continued)																		
RST	Restart	((SP) – 1) ← (PCH) ((SP) – 2) ← (PCL) (SP) ← (SP) – 2 (PC) ← 8 • (NNN)	1	1	*N*	*N*	*N*	1	1	1	1	3	11					
RZ	Return on Zero	If Z = 1, (PCL) ← ((SP)) (PCH) ← ((SP) + 1) (SP) ← (SP) + 2	1	1	0	0	1	0	0	0	1	1/3	5/11					
STACK, I/O, AND MACHINE CONTROL GROUP																		
DI	Disable Interrupts	The Interrupt system is disabled following the execution of the DI instruction.	1	1	1	1	0	0	1	1	1	1	4	·	·	·	·	·
EI	Enable Interrupts	The interrupt system is enabled following the execution of next instruction.	1	1	1	1	1	0	1	1	1	1	4	·	·	·	·	·
HLT	Halt	Processor is stopped; registers and flags are unaffected.	0	1	1	1	0	1	1	0	1	1	7	·	·	·	·	·
IN	Input	(A) ← (data)	1	1	0	1	1	0	1	1	2	3	10	·	·	·	·	·
NOP	No Operation	No operation is performed; registers and flags are unaffected.	0	0	0	0	0	0	0	0	1	1	4	·	·	·	·	·
OUT	Output	(data) ← (A)	1	1	0	1	0	0	1	1	2	3	10	·	·	·	·	·
POP B	Pop Registers B and C off Stack	(C) ← ((SP)) (B) ← ((SP) + 1) (SP) ← (SP) + 2	1	1	0	0	0	0	0	1	1	3	10	·	·	·	·	·
POP D	Pop Registers D and E off Stack	(D) ← ((SP)) (E) ← ((SP) + 1) (SP) ← (SP) + 2	1	1	0	1	0	0	0	1	1	3	10	·	·	·	·	·
POP H	Pop Registers H and L off Stack	(H) ← ((SP)) (L) ← ((SP) + 1) (SP) ← (SP) + 2	1	1	1	0	0	0	0	1	1	3	10	·	·	·	·	·
POP PSW	Pop Accumulator and Flags off Stack	$(CY) \leftarrow ((SP))_0$ $(P) \leftarrow ((SP))_2$ $(AC) \leftarrow ((SP))_4$ $(Z) \leftarrow ((SP))_6$ $(S) \leftarrow ((SP))_7$ (A) ← ((SP) + 1) (SP) ← (SP) + 2	1	1	1	1	0	0	0	1	1	3	10	‡	‡	‡	‡	‡
PUSH B	Push Registers B and C on Stack	((SP) – 1) ← (B) ((SP) – 2) ← (C) (SP) ← (SP) – 2	1	1	0	0	0	1	0	1	1	3	11	·	·	·	·	·
PUSH D	Push Registers D and E on Stack	((SP) – 1) ← (D) ((SP) – 2) ← (E) (SP) ← (SP) – 2	1	1	0	1	0	1	0	1	1	3	11	·	·	·	·	·
PUSH H	Push Registers H and L on Stack	((SP) – 1) ← (H) ((SP) – 2) ← (L) (SP) ← (SP) – 2	1	1	1	0	0	1	0	1	1	3	11	·	·	·	·	·
PUSH PSW	Push Accumulator and Flags on Stack	((SP) – 1) ← (A) $((SP) - 2)_0 \leftarrow (CY)$ $((SP) - 2)_1 \leftarrow 1$ $((SP) - 2)_2 \leftarrow (P)$ $((SP) - 2)_3 \leftarrow 0$ $((SP) - 2)_4 \leftarrow (AC)$ $((SP) - 2)_5 \leftarrow 0$ $((SP) - 2)_6 \leftarrow (Z)$ $((SP) - 2)_7 \leftarrow (S)$ (SP) ← (SP) – 2	1	1	1	1	0	1	0	1	1	3	11	·	·	·	·	·
SPHL	Move H and L to Stack Pointer	(SP) ← (H)(L)	1	1	1	1	1	0	0	1	1	1	5	·	·	·	·	·
XTHL	Exchange Top of Stack with H and L	(L) ↔ ((SP)) (H) ↔ ((SP) + 1)	1	1	1	0	0	0	1	1	1	5	18	·	·	·	·	·

condition flags and standard rules

There are five condition flags associated with the execution of instructions on the INS8080A. They are Zero, Sign, Parity, Carry, and Auxiliary Carry, and each flag is represented by a 1-bit register in the CPU. A flag is "set" by forcing the bit to 1, "reset" by forcing the bit to 0. The bit positions of the flags are indicated in the PUSH and POP PSW instructions.

Unless indicated otherwise, when an instruction affects a flag, it affects it in the following manner:

ZERO (Z): If the result of an instruction has the value 0, this flag is set; otherwise, it is reset.

SIGN (S): If the most significant bit of the result of the operation has the value 1, this flag is set; otherwise, it is reset.

PARITY (P): If the modulo 2 sum of the bits of the result of the operation is 0 (that is, if the result has even parity), this flag is set; otherwise, it is reset (that is, if the result has odd parity).

CARRY (CY): If the instruction resulted in a carry (from addition) or a borrow (from subtraction or a comparison) out of the high-order bit, this flag is set; otherwise, it is reset.

AUXILIARY CARRY (AC): If the instruction caused a carry out of bit 3 and into bit 4 of the resulting value, the auxiliary carry is set; otherwise, it is reset. This flag is affected by single-precision additions, subtractions, increments, decrements, comparisons, and logical operations; however, AC is used principally with additions and increments preceding a DAA (Decimal Adjust Accumulator) Instruction.

symbols and abbreviations

The following symbols and abbreviations are used in the subsequent description of the INS8080A instructions:

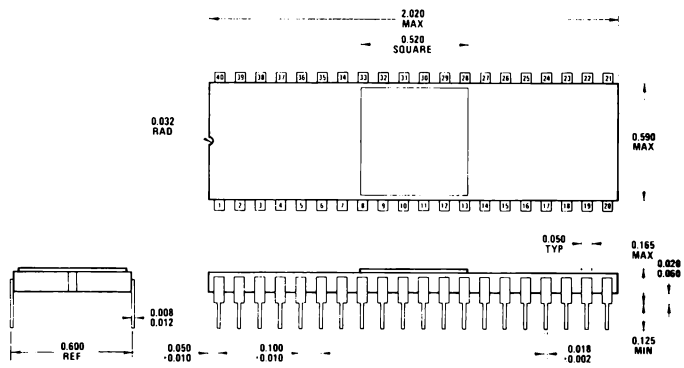
Symbols	Meaning
A	Register A (Accumulator)
B	Register B
C	Register C
D	Register D
H	Register H
L	Register L
DDD, SSS	The bit pattern designating one of the registers A, B, C, D, E, H, L (DDD = destination, SSS = source):

DDD or SSS	Register Name
111	A
000	B
001	C
010	D
011	E
100	H
101	L

byte 2	The second byte of the instruction
byte 3	The third byte of the instruction
port	8-bit address of an I/O device
r, r1, r2	One of the registers A, B, C, D, E, H, L

Symbols	Meaning
PC	16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8 bits respectively.)
SP	16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8 bits respectively.)
( )	The contents of the memory location or registers enclosed in the parentheses
←	"Is replaced by"
∧	Logical AND
∀	Exclusive OR
V	Inclusive OR
+	Addition
–	Twos complement subtraction
*	Multiplication
↔	"Exchange"
$\overline{\ }$	The ones complement (for example, $(\overline{A})$)
n	The restart number 0 through 7
NNN	The binary representation 000 through 111 for restart number 0 through 7 respectively
•	"Not affected"
0	"Reset"
1	"Set"
x	Unknown
↕	Flags affected according to Standard Rules

physical dimensions



Ceramic Dual-In-Line Package (D)
Order Number INS8080AD

Manufactured under one or more of the following U.S. patents: 3083262, 3189758, 3231797, 3303356, 3317671, 3323071, 3381071, 3408542, 3421025, 3426423, 3440498, 3518750, 3519897, 3557431, 3560765, 3566218, 3571630, 3575609, 3579059, 3593069, 3597640, 3607469, 3617859, 3631312, 3633052, 3638131, 3648071, 3651565, 3693248.

National Semiconductor Corporation
2900 Semiconductor Drive, Santa Clara, California 95051, (408) 737-5000/TWX (910) 339-9240
National Semiconductor GmbH
808 Fuerstenfeldbruck, Industriestrasse 10, West Germany, Tele. (08141) 1371/Telex 05-27649
National Semiconductor (UK) Ltd.
Larkfield Industrial Estate, Greenock, Scotland, Tele. (0475) 33251/Telex 778-632

```
110 0039 0100 A  INBUF:  .WORD   IBUF              INPUT BUFFER POINTER
111 003A 8000 A  SEND:   .WORD   X'8000            ADDRESS OF UAR/T (SEND)
112 003B 8000 A  RECV:   .WORD   X'8000          ; ADDRESS OF UAR/T (RECEIVE)
113 003C 8001 A  DRRST:  .WORD   X'8001          ; RESET ADDRESS FOR DR
114 003D 8002 A  MRST:   .WORD   X'8002          ; MASTER RESET ADDRESS
115 003E 8003 A  LBAUD:  .WORD   X'8003          ; BAUD RATE LOAD ADDRESS
116 003F 0019 A  BAUDR:  .WORD   X'19            ; CONSTANT-1200 BAUD,
117                                              ;   EVEN PARITY.
118 0040 0200 A  RST01:  .WORD   OUTBUF          ; CONSTANTS TO RESET OUTPUT
119 0041 0100 A  RST02:  .WORD   256             ;   BUFFER POINTER, COUNTER
120 0042 0311 A  XDONE:  .WORD   XMITC           ; POINTER FOR OUTPUT COMPLETE
121 0043 007F A  MASK:   .WORD   X'7F            ; MASK FOR ASCII CHARACTERS
122
123
124      0100            .=0100
125
126
127
128                      ;*****************************************************
129
130
131                      ;  INPUT DATA BUFFER
132
133      0200    IBUF:   .=.+256
134
135                      ;*****************************************************
136
137
138                      ;  OUTPUT DATA BUFFER
139
140      0300    OUTBUF: .=.+256
141
142                      ;*****************************************************
143
144
```

Figure 5. UART-Serial I/O via UAR/T

MICROPROCESSOR CONSULTANTS

COMPUTERWISE
13126 South 71 Hwy.
Kansas City, Missouri 64030
(816) 765-3330
Contact: Bill Brown

BINARY ENTERPRISES
24794 Greenfield
South Bend, Indiana 46628
(219) 277-0691

INTERPHASE ASSOCIATES
619 Newberry Drive
Richardson, Texas 75080
(214) 231-4459
Contact: Mike Cope

SHEPARDSON MICROSYSTEMS
Suite 302, 10601 S. Saratoga-Sunnyvale Road
Cupertino, California 95014
(408) 257-2996

MICROCOMPUTER CONCEPTS, INC.
10683 Cranks Road
Culver City, California 90230
Formerly: C.A. Pullen & Assoc.

COMPUWRAP
10357 Kinsman Road
Newbury, Ohio 44065
(216) 564-1152

SUDBURY SYSTEMS, INC.
80 Union Avenue
Box Q
Sudbury, Mass. 01776
(617) 443-3133

DANYL CORPORATION
310 Cooper Center
North Park Dr. &
Browning Road
Pennsauken, N.J. 08109
(609) 662-6615

GILL STARR MICRO-DIGITAL, INC.
2357 Lemoine Avenue
Ft. Lee, N.J. 07024
(201) 461-3324

SYSCOM
3058-B Scott Blvd.
Santa Clara, Ca. 95050
(408) 246-2437

ABLER DATA SYSTEMS, INC.
740 Garvens Avenue
Brookfield, Wisconsin 53005
(414) 786-2448

Subroutine Library Catalog

SL0001A, IMP-16
BINBCD
Binary to BCD Conversion — BINBCD converts a binary number to its equivalent BCD number, using the standard IMP-16 Instruction Set.

SL002A, IMP-16
BCD
Binary to BCD Conversion and BCD to Binary Conversion — BCD consists of 2 subroutines. The first converts a binary number to its equivalent BCD number. The second converts a BCD number to its equivalent binary number. Both routines use IMP-16 Extended Instructions.

SL0003A, IMP-16
MD
Multiply and Divide Routines — MD contains divide and multiply routines compatible to the routines available in the IMP-16 CROM-2 Extended Instruction Set. From outward appearances they will be identical except for execution time.

SL0004A, IMP-16
PTBIN (may have errors)
Binary Tape Punch and Verify Package — PTBIN is package of 2 main programs and 6 subroutines for the IMP-16P.

PTPNCH	— Binary Paper Tape Punch.
SEARCH	— Loads PTPNCH format Binary Tape.
LEADER	— Punches 40 null characters on tape.
CKSM	— Reads, packs, and stores ASCII characters.
ASCBIN	— Converts 4 packed hex ASCII chars to Binary and stores in AC1.
UNPACK	— Unpack ASCII chars, STRIP parity bit and right adjust in AC0, AC1.
READPK	— Reads 2 ASCII chars from TTY and pack into AC0.

SL0005A, IMP-16
BINASC
Binary to HEX ASCII Conversion — BINASC converts a binary number in AC0 to 4 hex ASCII characters and prints the result on the TTY using the PUTC routine in TTY16P.

SL0006A, IMP-16
BINGRAY
Binary to Gray Code Conversion — Converts up to 16 binary numbers to their equivalent Gray Code representation and stores the result in a 16-word table.

SL0007A, IMP-16
BCDBIN
BCD to Binary Conversion — Converts 4 unpacked BCD digits to its binary equivalent using the standard IMP-16 Instruction Set.

SL0008A, IMP-16
PNMULT
Positive/Negative Conversion Routine — PNMULT accepts positive and/or negative 16-bit numbers to be used by MD (SL0003A). Converts to positive before the multiply or divide operation and assigns a sign to the product based on the signs of the operands.

SL0009A, IMP-8

	IMP-8 Math Routines
MPY	Unsigned Multiply
SMPY	Signed Multiply
DIV	Unsigned Divide
SDIV	Signed Divide
BCDBIN	BCD to Binary Conversion
BINBCD	Binary to BCD Conversion
DADD	Double Precision Addition
DSUB	Double Precision Subtraction
DCMP2	Double Twos Complement
RANDOM	Random Number Generator
PWR	I to the power of J
SQRT	Integer Square Root

SL0010A, IMP-16
MEMORY DUMP
A general memory dump program to allow any section of memory to be dumped through TTY keyboard access. There are no restrictions as to dump range or dump locations. (Contributed by Crest Engineering, Minneapolis.)

SL0011A, IMP-16
GALPAT
This is an IMP-16 memory diagnostic diagnostic designed to be loaded into two 256 X 8-bit PROMs located on an IMP-16C card in the FF00-FFFF slots. It exercises memory locations 0000-0FFF, and uses only the hardware stack and registers for data storage.

SL0012A, IMP-16
RAMDUMP
This program punches out data from RAM or PROM. Standard RLM format is used. The program prompts for the start and end addresses. It replaces SL0004A, PTBIN.

SL0013A, IMP-16
TAPE TITLER
This program develops alpha-numeric characters in a 6 X 8 dot format and prints them longitudinally on the tape. The characters are then readable with the eye. It can be used in conjunction with the assembler and will punch up to 60 characters.

SL0014A, IMP-16
GRAY CODE CONVERSION
This routine converts 8 or 16 bit GRAY CODE to binary.

SL0015A, PACE
PACRAM
This program has commands for punching paper tapes, reading paper tapes, punching leader, executing a program and altering memory locations.

Note: The punch format is absolute binary. (Contributed by TELEPLEX.)

SL0016A, IMP-16
PRTPLT
Print Plot routines uses TTY or High Speed Printer. (Contributed by NASA/AMES.)

SL0017A, IMP-16
TSTPLT
Data for testing SL0016A.

SL0018A, PACE
CALCULATOR
This program makes a PACE Microprocessor Development System, with a Teletype-type terminal, into a nine decimal digit, four-function calculator, using only positive integers of nine or fewer digits. There is no check for incorrect results (overflow) of addition or multiplication, and no check for negative results from subtract, but in these cases, the output routine may halt in the double precision divide routine. The

division routine calculates a double precision quotient and a double precision remainder, so there is no error situation. (0/0 = 0,0)

In operation; this program prints a > (greater than) sign at the left margin, and waits for you to enter nine or fewer decimal digits, without any other characters or blanks, one arithmetic operation symbol (+ – * /), a second operand and any character (the equal sign looks best, and a carriage return will cause overprinting); then the program calculates the rightmost 32 bits of the sum, difference, product or the 32 bit quotient and the 32 bit remainder, and prints the result(s).

Examples:
5 + 37 = 000000042
37 – 5 = 000000032
>2*2 = 000000004
>5/2 = 000000002, 000000001

This program can run as a subroutine under the DEBUG program (PACDBG).

(Contributed by C. Strain, ELECTRO Units Corp.)

SL0019A, IMP-16
MESGH
Output Routine for High Speed Printer analogous to TTY MESG Subroutine.

SL0020A, IMP-16
CHARST
Character String Routines for IMP-16 (Veripen Corp.).

SL0026A, IMP-16, PACE
TABTAP
This program reduces the length of the source tapes from the IMP and PACE Editors by using a single tab character to replace a string of spaces. This program overwrites the card reader routines in the editors and may be loaded by pressing 'Load Prog' again after the editor is loaded.

SL0027A, SC/MP
SC/MP Match Package
SC/MP Math Routines. Double Add, Double Negate, Double Subtract, Multiply, Divide, BCD Multiply, BCD Add, BCD Subtract, BCD Complement, BCD Divide.

SL0028A, IMP-16
SQRT
Takes a 24-bit square root of a 32-bit number.

SL0021A, IMP-16
CONTAP
Converts an RLM tape to a printed listing that is useful for punching RLM cards.

SL0022A, PACE
NUMPRG
Converts a binary value in register 0 to a six character ASCII value in registers 0, 1, and 2.

SL0023A, IMP-16
Disc RLM-Promsft-B
Update for Promsft-B. Allows PROM to be programmed using a RLM from the disc.

SL0024A, IMP-16
Disc RLM-Promsft-C
Update for Promsft-C. Like SL0023A.

SL0025A, PACE
PALM
PALM 'Punch Absolute Load Module'. PALM punches an absolute Load Module of any selected memory. An entry location may also be specified for program execution. Blocks of 12 are always punched and the specified end of range will be extended to accommodate this block of 12 words.

Note: PALM will even punch itself.

TTY Prompts: TYPE IN THE RANGE *0123:ABCF*
Response is four hex digits, colon, four hex digits.

TTY Prompts: ENTRY *045F (turn on punch)*
Response is four hex digits and turn on punch.

Load Module is punched and the system will halt.

SC/MP Application Cards

Bob Pecotich, National Semiconductor

The SC/MP family of microprocessor application cards are here **NOW.** Products are:

CPU CARDS
ISP-8C/100 – card with on-card timing, buffered 16-bit address, and 8-bit data bus, includes 256 bytes of RAM and sockets for 512 bytes of PROM/ROM (MM5204Q/MM5214).

MEMORY CARDS
ISP8C/002 – 2k byte static read/write memory card, with on-card address decode.

ISP-8C/004B or P – 4k byte ROM/PROM memory card. "B" designates card with sockets for 8 MM5204Q/MM5214; "P" designates card with 8 blank MM5204Q's.

Users are (among others):

Engineers replacing electro-mechanical controls and logic with a purely electronic approach, e.g., pipeline, engine, and complex appliance controls.

Medical Analysis/Diagnostic System Manufacturers.

Electronic Lab Measurement Systems Manufacturers.

Broad categories of users having limited electronic engineering, PC card fab or, software experience, but wanting to implement microprocessor cost savings.

SC/MP Cards offer:

Ease of use/system integration/software development. Low use cost. Look at the prices!

	1	10	25	50	100
ISP-8C/100	$250	$238	$218	$188	$ 98
ISP-8C/002	160	152	139	120	99
ICP-8C/004B	125	119	109	94	60
ISP-8C/004P	525	499	457	394	325

Class Schedules

	Eastern Training Center	Western Training Center
Microprocessor Fundamentals	November 8-12	October 25-29
SC/MP Applications	November 15-10	November 8-12
PACE Applications	October 18-22	November 1-5

Processor is Calculator-Oriented!

Looking for a versatile, low-cost, dedicated or custom-programmable calculator or control system? We've got it! Our MM5799 contains all system timing functions, all arithmetic and logic functions, all RAM functions (384 bits), and all control ROM functions (1536 microinstructions 8-bits wide, 10-μs/microcycle) that you'll need to implement a variety of small control and microprocessor systems.

A single MOS/LSI chip, the MM5799 can scan 56 keyboard switches, or you can enter BCD data words. Its eight outputs present information in either a BCD or a seven-segment-plus-decimal-point format, and four additional latched outputs give you encoded digit-timing information. Further, a serial-in port and a serial-out port let you expand the basic RAM store and interface to peripherals.

And speaking of peripherals and extra storage, our MM5788 printer interface, DS8664 Series oscillator and decoder/drivers, MM5785 RAM interface, and MM2102 and MM74C930 1k static RAMs are a perfect match to an MM5799-based system.

For information, contact National Semiconductor Marketing Services, 2900 Semiconductor Drive, Santa Clara, CA 95051.

Support Hardware for SC/MP and PACE Application Cards!

Bob Pecotich, National Semiconductor

SC/MP and PACE share a common 4.375" X 4.862" card size, with both families of application cards using the standard 72 pin, 0.1 with center edge connector. The card size is one widely supported by AUGAT, INC. (with its M Series line) and a number of second sources. The following table lists support hardware commonly needed:

Manufacturer	Part No.	Price/Qty.
72 CONTACT EDGE CONNECTOR		
Augat	14005-17P3	$4.81/1, $4.22
Robinson/Nugent	EC721	$7.55/1, $5.20/100
Elco	00-6307-072-309-001	
Cerich	50-72C-30	
National Connector	900100-36	
Stanford Applied Engineering	CDP7000-72	
Winchester	HW36C0111	$4.79/1, $3.51/100
CONNECTOR CARD CAGE WITH BACKPLANE:		
Augat	8170-MG1	$177.50/1, $147.75/10-24
Robinson/Nugent	MECA-1	$202/1, $150/10-24
Augat	8136-MG13	$29.50/1, $24.50/10-24
Robinson/Nugent	EB-72	$31.60/1
UNIVERSAL W/W CARD WITH TERMINALS:		
National Semiconductor	IPC-16C/801	$40.00/1

NEW PROGRAMS

IMP-16 SQRT Program — SL0028A source listing only
Contributed by: Microcomputer Concepts, Inc.,
10683 Cranks Road, Culver City, CA 90230
(212) 836-2271

```
SQRT    SQUARE ROOT                          PAGE NUMBER 0001


                        .TITLE  SQRT,'SQUARE ROOT'
                        .EXTD
                NEG = 11
                POS = 2
                ZERO = 1
                NZERO = 5
                CYOV = 10
                R0 = 0
                R1=1
                R2=2
                R3=3
                        .=.+02F00
                        .PAGE   'SQUARE ROOT'
                        .LOCAL
                ;##########################################
                ;SQUARE ROOT
                ;INPUT:  R0,R1 = X
                ;OUTPUT:  R0,R1 = SQRT (X)
                ;TEMPORARIES USED:  FA-FF
                ;SUBROUTINES CALLED:  NONE
                ;STACK USAGE:  1
                ;##########################################
                PSR=0FA
                SQU=0FC
                TEMP=0FE
        SQRT:
2F00 8935 A             LD      R2,LPSR
2F01 8D2F A             LD      R3,X2000 ;BIT 13
2F02 A0FC A             ST      R0,SQU  ;DEVELOPING SQUARE
2F03 A4FD A             ST      R1,SQU+1
2F04 4C00 A             LI      R0,0
2F05 4D00 A             LI      R1,0
2F06 A0FA A             ST      R0,PSR
2F07 A0FB A             ST      R0,PSR+1        ;CLEAR PARTIAL (DEVELOPING) SQUARE ROO
        $0:
                ;R0,R1 = PSR            ;SET SELFF FOR DP LEFT SHIFT USING LINK
2F08 0A00 A             SFLG    2
2F09 5D01 A             SHL     R1,1
2F0A 5801 A             ROL     R0,1
2F0B F92A A             SKNE    R2,LPSR ;SKIP DURING LEAST SIG PORTION
2F0C 2102 A             JMP     $0A
2F0D 3D82 A             RXOR    R3,R1   ;LS
2F0E 2101 A             JMP     .+2
        $0A:
2F0F 3C82 A             RXOR    R3,R0   ;MS
2F10 A0FE A             ST      R0,TEMP
2F11 A4FF A             ST      R1,TEMP+1
2F12 80FC A             LD      R0,SQU
2F13 84FD A             LD      R1,SQU+1
2F14 B4B0 A             DSUB    TEMP    ;SQU - TRIAL
2F15 00FE A
                                        ;SELFF = 0
2F16 120F A             BOC     POS,$4
2F17 80FC A             LD      R0,SQU  ;RESTORE SQU
2F18 84FD A             LD      R1,SQU+1
2F19 2911 A             JSR     SHAST
                        .PAGE
                ;CHECK FOR DONE
        $1:
2F1A 80FA A             LD      R0,PSR
2F1B 84FB A             LD      R1,PSR+1
2F1C 0A80 A             PFLG    2
2F1D 5BFF A             ROR     R3,1
2F1E F917 A             SKNE    R2,LPSR ;SKIP DURING LEAST SIGNIFICANT PORTION
2F1F 2103 A             JMP     $2
2F20 FD14 A             SKNE    R3,C0040        ;STOP AT 24 BITS
2F21 2109 A             JMP     SHAST   ;DONE
2F22 21E5 A             JMP     $0      ;LESS THAN 24 BITS DONE
                ;
        $2:
2F23 FD10 A             SKNE    R3,X8000
2F24 4A01 A             AISZ    R2,1    ;NEVER SKIPS
2F25 21E2 A             JMP     $0
                ;SET ROOT BIT
        $4:
2F26 2904 A             JSR     SHAST   ;SHIFT R0,R1 LEFT; STORE IN SQU
2F27 3C81 A             RCPY    R3,R0   ;SET ROOT BIT
2F28 6A80 A             OR      R0,0(R2)
2F29 A200 A             ST      R0,0(R2)
2F2A 21EF A             JMP     $1
                ;
                ;
        SHAST:
2F2B 0A00 A             SFLG    2
2F2C 5D01 A             SHL     R1,1
2F2D 5801 A             ROL     R0,1
2F2E A0FC A             ST      R0,SQU
2F2F A4FD A             ST      R1,SQU+1
2F30 0200 A             RTS
                ;
                ;
2F31 2000 A X2000:      .WORD   02000
2F32 0000 A X0000:      .WORD   00000
2F33 0001 A C0001:      .WORD   00001
2F34 8000 A X8000:      .WORD   08000
2F35 0040 A C0040:      .WORD   00040
2F36 00FA A LPSR:       .WORD   PSR
0000                .END

C0001  2F33  T#  C0040  2F35  T  CYOV   000A  A#  LPSR   2F36  T  NEG    000B  A#
NZERO  0005  A#  POS    0002  A  PSR    00FA  A   R0     0000  A  R1     0001  A
R2     0002  A   R3     0003  A  SHAST  2F2B  T   SQRT   2F00  T#  SQU    00FC  A
TEMP   00FE  A   X0000  2F32  T#  X2000  2F31  T   X8000  2F34  T  ZERO   0001  A#
$0     2F08  T   $0A    2F0F  T   $1     2F1A  T   $2     2F23  T  $4     2F26  T
NO ERROR LINES
```

MOVING ON?

Let us know, complete the form below and send to:

UNITED STATES
Marketing Services/520
National Semiconductor
2900 Semiconductor Drive
Santa Clara, CA 95051

GERMANY
National Semiconductor GmbH
808 Fuerstenfeldbruck
Industriestrasse 10
Tel: 08141/1371
Telex: 05-27649

NEW ADDRESS

NAME _____

TITLE _____

COMPANY _____

ADDRESS _____

CITY _____

STATE _____ ZIP _____

DATE EFFECTIVE _____

ADDRESS LABEL OR PRINT OLD ADDRESS

NAME _____

TITLE _____

COMPANY _____

ADDRESS _____

CITY _____

STATE _____ ZIP _____

LIFETIME MEMBERSHIP FORM!

Lifetime membership fee is only $15.00. Make checks payable to COMPUTE and send to:

UNITED STATES
COMPUTE/208
National Semiconductor
2900 Semiconductor Drive
Santa Clara, CA 95051

GERMANY
National Semiconductor GmbH
808 Fuerstenfeldbruck
Industriestrasse 10
Tel: 08141/1371
Telex: 05-27649

NAME _____

TITLE _____

COMPANY _____

ADDRESS _____

CITY _____

STATE _____ ZIP _____

User Library Order Form

PROGRAM		NAME	LISTING QUANTITY N/C	SOURCE PAPER TAPES QUANTITY $5.00 EACH
SL0001A	IMP	BINBCD		N/A
SL0002A	IMP	BCD		N/A
SL003A	IMP	MD		N/A
SL0004A	IMP	PTBIN		N/A
SL0005A	IMP	BINASC		N/A
SL0006A	IMP	BINGRAY		N/A
SL0007B	IMP	BCDBIN		N/A
SL0008A	IMP	PNMULT		N/A
SL0009A	IMP	IMP-8 MATH		N/A
SL0010A	IMP	MEMORY DUMP		
SL0011A	IMP	GALPAT		
SL0012B	IMP	RAMDUMP		
SL0013A	IMP	TAPE TITLER		
SL0014A	IMP	GRAY CODE		
SL0015A	PACE	PACRAM		
SL0016A	IMP	PRTPLT		
SL0017A	IMP	TSTPLT		
SL0018A	PACE	CALCULATOR		
SL0019A	IMP	MESGH		
SL0020A	IMP	CHARST		
SL0021A	IMP	CONTAP		
SL0022A	PACE	NUMPRG		
SL0023A	IMP	DISC RLM-PROMSFT-B		
SL0024A	IMP	DISC RLM-PROMSFT-C		
SL0025A	PACE	PALM		
SL0026A	PACE IMP	TABTAP		
SL0027A	SC/MP	SC/MP MATH PACKAGE		
SL0028A	IMP	SQRT		

Fill out the form completely, make your check payable to COMPUTE, and mail to:

UNITED STATES
COMPUTE/208
National Semiconductor
2900 Semiconductor Drive
Santa Clara, CA 95051

GERMANY
National Semiconductor GmbH
808 Fuerstenfeldbruck
Industriestrasse 10
Tel: 08141/1371
Telex: 05-27649

NAME _____

TITLE _____

COMPANY _____

ADDRESS _____

CITY _____

STATE _____ ZIP _____

Please make sure the programs you select are for the microprocessor you have.

UNITED STATES

COMPUTE/208
NATIONAL SEMICONDUCTOR CORP.
2900 SEMICONDUCTOR DRIVE
SANTA CLARA, CA. 95051
TEL: (408) 247-7924
TWX: 910-338-0537

EUROPE

GERMANY
National Semiconductor GmbH
808 Fuerstenfeldbruck
Industriestrasse 10
Tel: 08141/1371
Telex: 05-27649

AUSTRALIA
NS Electronics Pty Ltd
Cnr. Stud Rd. & Mtn. Highway
Bayswater, Victoria 3153
Tel: 03-729-6333
Telex: 32096