

**32 BIT MICROPROCESSOR SYSTEM**  
**module Z3/EV**

**Volume 2/4**

**THEORY, EXERCISES AND APPENDIX**

***TEACHER/STUDENT manual***





**INDEX**

<b>Lesson F43:</b>	<b>SERIAL INTERFACE</b>	<b>Page. 1</b>
<b>Lesson F44:</b>	<b>DIGITAL/ANALOG CONVERSION</b>	<b>Page. 17</b>
<b>Lesson F45:</b>	<b>ANALOG/DIGITAL CONVERSION</b>	<b>Page. 24</b>
<b>Appendix A:</b>	<b>Commands to the Monitor of Module Z3/EV</b>	<b>Page. 33</b>
<b>Appendix B:</b>	<b>Resources of the Monitor of Module Z3/EV</b>	<b>Page. 40</b>
<b>Appendix C:</b>	<b>Communications with the PC</b>	<b>Page. 47</b>
<b>Appendix D:</b>	<b>Datasheet Microprocessor 80386EX</b>	<b>Page. 51</b>
<b>Appendix E:</b>	<b>Summary of Instructions 80386EX</b>	<b>Page. 65</b>
<b>Appendix F:</b>	<b>Electrical Diagrams</b>	<b>Page. 96</b>

---

## SAFETY RULES

Keep this handbook at hand for any further help.

After the packaging has been removed, set all accessories in order so that they are not lost and check the equipment integrity. In particular, check that it shows no visible damage.

Before connecting the equipment to the power supply, be sure that the rating corresponds to the one of the power mains.

**This equipment must be employed only for the use it has been conceived, i.e. as educational equipment, and must be used under the direct supervision of expert personnel.**

Any other use is not proper and therefore dangerous. The manufacturer cannot be held responsible for eventual damages due to inappropriate, wrong or unreasonable use.

---

## LESSON F43: INTERFACE SERIAL

### OBJECTIVES

- Principles of serial communication
- Serial Interface of Module Z3/EV
- The controller 8250
- The registers of the controller 8250
- The pin of the controller 8250
- Programming of the serial interface
- Development of application programs and exercises.

### MATERIALS

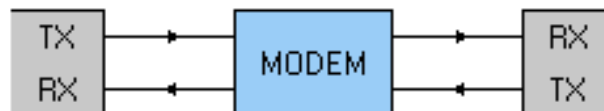
- Base unit for the IPES system (power supply mod.PS1-PSU/EV, module holder mod.MU/EV, individual management unit mod.SIS1/SIS2/SIS3)
- Experimentation module mod. Z3/EV
- oscilloscope.

### F43.1 PRINCIPLES of SERIAL COMMUNICATION

The serial transmission of data between computers is used, instead of the parallel one, when:

- the distance is greater than 3-4 m
- the numbers of the connection lines has to be minimal.

The serial transmission makes possible the transmission of data between systems, which are located at big distances from each other, using only a few lines. it also makes possible the use of the telephone line using special modulation devices, called Modem (see Fig. F43.1).



*fig. F43.1*

Since the microprocessor works with a data bus of the parallel type, and normally, the minimum quantity is the byte, the need for data conversion from parallel to serial is born.

Shift Registers are used, as shown in Fig. F43.2.

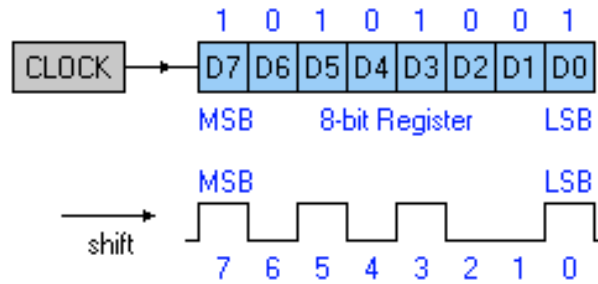


fig. F43.2

### F43.2 DATA FORMAT in the ASYNCHRONOUS SERIAL COMMUNICATION

In order to transmit data in the asynchronous mode; that is, without a common clock between who transmits and who receives, over a serial line, special information has to be added to the bit of the byte being transmitted.

A particular format for standard transmission is born, shown in Fig. F43.3, where:

- the state of the transmission line is normally 'high (logic level 1)
- the transmission begins with the **Start Bit** at logic level 0
- then, the **character bit** to be transmitted are sent in line, starting with the least significant (the number of bit can be 7 or 8, and the polarity is kept)
- after the character bit it's possible to add a **parity bit**, which can be:
  - Even: is set to 1 or to 0 to make the total addition of bit '1' be even
  - Odd: is set to 1 or to 0 to make the total addition of bit '1' be odd
- at the end of the transmission a **Stop Bit** is added, which can be 1 or 2 bit long
- The duration of the single bit depends from the speed of transmission used (**baud-rate**) which normally assumes the following values:  
300, 600, 1200, 2400, 4800, 9600, 19200, 38400, ..

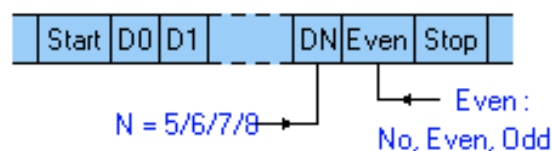


fig. F43.3

With respect the levels of the signals on line, the following rules apply:

- one signal between +3V e +12V represents the logic value '1'
- one signal between -3V e -12V represents the logic value '0'
- one signal between -3V e +3V is in a non defined logic state

*All these rules define that which goes under the name Standard RS-232 for the Asynchronous Serial Transmission.*

### F43.3 SERIAL INTERFACE of MODULE Z3/EV

The serial interface of Module Z3/EV satisfies all the characteristics of the standard RS-232.

In particular, also the signals available on the output connector J2 respect this standard.

*A disposition of the compatible pins with the 9-pin connector of the serial interface of the personal computer has been chosen.*

The diagram of the serial interface is shown in figure F43.4 .

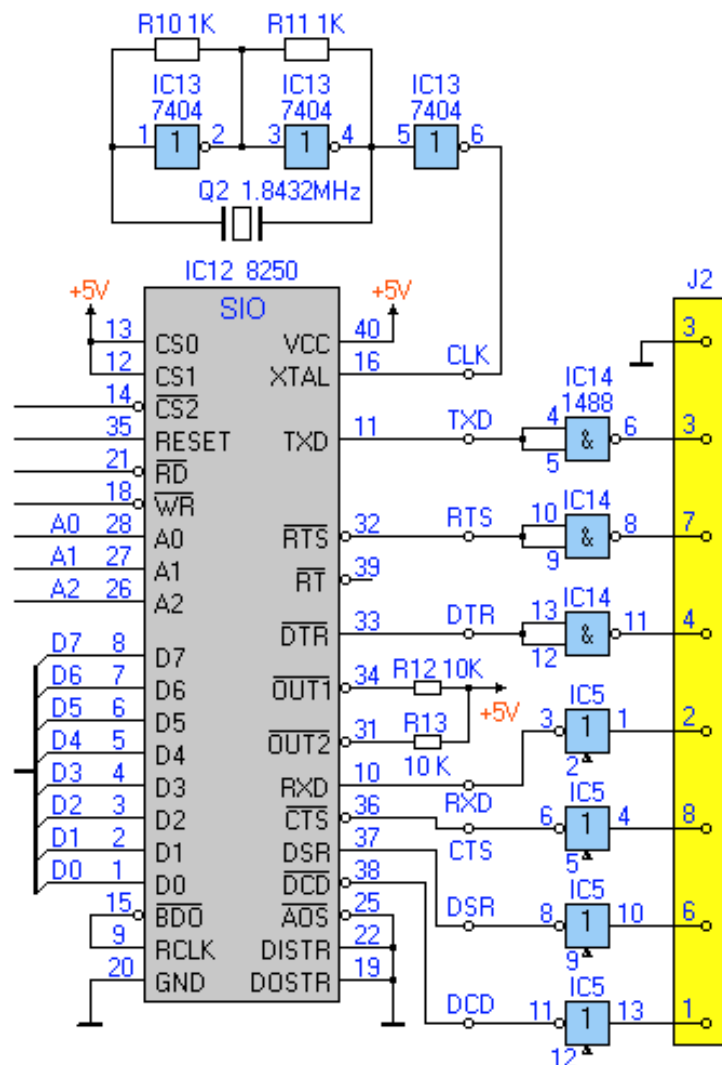


fig. F43.4

The serial interface uses the Asynchronous Communication Controller 8250.

It is a device of programmable I/O, of general use, projected for the Intel microprocessors.

An external oscillator at 1.8432 MHz provides the transmission and reception clock. This clock is then divided internally of the 8250 to furnish the different baud rate.

The signals present on the output connector J2 are:

- Pin 1 (DCD): Data Carrier Detect (input signal). It is a management signal for the Modem.
- Pin 2 (RXD): Received Data (input signal). Serial data received from the transmitting external device.
- Pin 3 (TXD): Transmitted Data (output signal). Serial data transmitted to the receiving external device.
- Pin 4 (DTR): Data Terminal Ready (output signal). Modem management signal that serves to communicate that the Module Z3 is ready to begin a communication session.
- Pin 5 (GND): Ground.
- Pin 6 (DSR): Data Set Ready (input signal). Modem management signal which serves the external device to indicate the Module Z3 that it is ready to begin a communication session.
- Pin 7 (RTS): Request To Send (output signal). Indicates the requirement of the Module Z3 for the transmission of a byte.
- Pin 8 (CTS): Clear To Send (input signal). Indicates the requirement, of the external device, for the transmission of a Byte.

The controller 8250 works with signals at logic level TTL, while for the output signals, the standard RS-232 requires levels of +/-12V. The CI IC14 (1488) and IC15 (489) are used for changing levels.

#### **F43.4 THE CONTROLLER 8250: PERFORMANCE and INTERNAL REGISTERS**

The controller 8250 is a peripheral, of the Intel family, which provides the complete implementation of the standard RS-232. In particular:

- programmable internal clock divider for furnishing baud-rates from 75 to 9600 b/s
- start bit, stop bit, bit parity programming
- input and output double buffer
- input for a clock of independent
- modem signals control: CTS, RTS, DSR, DTR.

The different functioning modes of the controller are selected by programming the internal registers.

It communicates with the microprocessor by means of the following addresses, which correspond to as many internal registers:



- 330H      Receive Buffer Register
- 330H      Transmit Holding Register
- 333H      Line Control Register
- 335H      Line Status Register
- 334H      Modem Control Register
- 336H      Modem Status Register
- 330H      Divisor Latch LSB
- 331H      Divisor Latch MSB
- 332H      Interrupt Identification Register
- 331H      Interrupt Enable Register

### **F43.5 THE CONTROLLER 8250: EXTERNAL SIGNALS**

With reference to figure F43.4, the controller 8250 is provided with the following I/O lines:

#### **Input Signals.**

- CS0, CS1, CS2# (Chip Select): chip-enabling signals. It is used only CS2#.
- DISTR# (Data Input Strobe): When DISTR# is low, the chip is selected and reading is possible.
- DOSTR# (Data Output Strobe): When DOSTR# is low, the chip is selected and writing is possible.
- ADS# (Address Strobe): When it's low, the chip reads the addresses from A0,A1,A2.
- A0,A1,A2 (Register Select): address lines for the selection of the registers. They arrive from the analog lines of the microprocessor's bus.
- RESET (reset): initializes the processor.
- RCLK (Receiver Clock): reception clock
- RXD (Received Data): serial input of data coming from the communication line.
- CTS# (Clear To Send): modem control line for the transmission of data, When it's active, data can be transmitted.
- DSR# (Data Set Ready): modem control line to determine the communication.
- DCD# (Data Carrier Detect): is active when the communication is active.
- RI# (Ring Indicator): not used.

#### **Output signals.**

- DTR# (Terminal Ready): When it's active indicates that the 8250 is ready to communicate.
- RTS# (Request To Send): When it's active indicates that the 8250 is ready to transmit a byte.
- OUT1# (Output1): not used.
- OUT2# (Output2): not used.
- BAUDOUT# (Baud Out): transmission clock.

- TXD (Transmitted Data): serial output for the data to be sent over the communication lines.

#### **Input/Output Signals.**

- D0-D7 (Data Bus): data bus for the microprocessor connection.
- XTAL (Clock Input): input for the clock.

### **F43.6 PROGRAMMING of the SERIAL INTERFACE**

For the writing of communication programs it's possible to use directly the registers of controller 8250.

This mode of proceeding results very complex since the quantity of information to be handled is high.

To simplify the programming of the serial interface, the monitor of the system contains a software interruption ( INT 14H ) with all the functions foreseen for the serial communication.

#### **INT 14H: Management of the Serial Interface.**

This interruption drives the of the asynchronous serial interface RS-232. the available functions are determined from the contents of AH:

<b>INPUT</b>	
	AH = 0 Initialization
	AH = 1 Sending character
	AH = 2 Receiving character
	AH = 3 Reading State
	AH = 4 Modem Control

#### **AH = 0 : Initialization of communication port**

The contents of AL determines the parameters of the initialization:

<b>Register AL</b>	<b>Programming</b>
Bit 7,6,5	Baud rate = 000 : 1200 = 001 : 2400 = 010 : 4800 = 011 : 9600
Bit 4,3	Parity = 00 : no = 01 : odd = 10 : no = 11 : even
Bit 2	Stop Bit = 0 : 1 = 1 : 2
Bit 1,0	Length word = 10 : 7 bits = 11 : 8 bits

**AH = 1 : Transmission of a character**

The contents of AL determines the character to send.

Before sending the character, the routine waits for the other eventual preceding characters to be transmitted.

**AH = 2 : Reception of a character**

The contents of AL determines the received character.

The routine waits for the availability of a character before returning to the calling program.

**AH = 3 : Reading of the state**

The contents of AH determines the present state of the line and of the modem.

Register AH	State of Modem
Bit 7	Received line signal detect
Bit 6	Ring indicator
Bit 5	Data set ready
Bit 4	Clear to send
Bit 3	Delta receive line signal detect
Bit 2	Trailing edge ring detector
Bit 1	Delta data set ready
Bit 0	Delta clear to send
Register AL	State of the Line
Bit 7	Time-out
Bit 6	Transmitter shift register empty
Bit 5	Transmitter holding register empty
Bit 4	Break detect
Bit 3	Framing error
Bit 2	Parity error
Bit 1	Overrun error
Bit 0	Data ready

**AH = 4 : Control of the Modem**

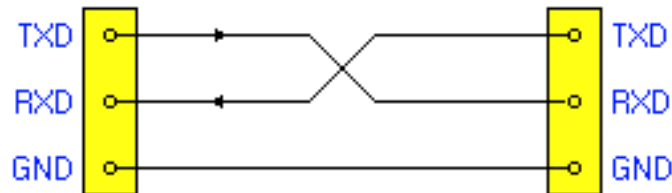
The contents of AL determines the present state of the Modem that will be set.

Register AL	State of the Modem
Bit 7	0
Bit 6	0
Bit 5	0
Bit 4	Loop
Bit 3	Out2
Bit 2	Out1
Bit 1	Request To Send (RTS)
Bit 0	Data Terminal Ready (DTR)

### F43.7 CONNECTION between MODULE Z3 and a REMOTE SERIAL UNIT

There are different possibilities to connect between each other, two units for the serial communication.

The simplest one, foresees a connection with 3 cables of the type shown in Fig. F43.5.



*fig. F43.5*

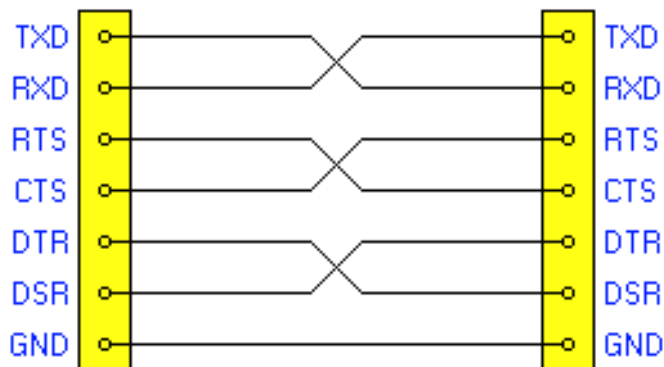
This type of connection shows the problem of the lack of synchronization signals between the two devices, and therefore, lack of control of the data flow.

The transmitting device can begin the transmission before the receiving device is ready, therefore with loss of data.

Otherwise, if the transmitting device operates faster than that which receives, there will be a loss of data (byte) during the communication.

Normally, communication lines are used to control the data flow between the two devices, with the finality of avoiding any loss of information.

The connection with control signals of the data flow is shown in Fig. F43.6.



*fig. F43.6*

- The lines DTR (Data Terminal Ready) and DSR (Data Set Ready) are used to make the connection between the two devices. They are activated at the beginning of the communication session, and inactivated at the end.
- The lines RTS (Request To Send) and CTS (Clear To Send) are used instead to control the transmission of the single byte.

See the example on the use in the experimentation, for the details.

**F43.8 EXERCISES and SUMMARY QUESTIONNAIRE**

➡ <i>Z3</i>	
➡ <i>SIS1</i>	Set all switches in the <i>OFF</i> position
➡ <i>SIS2</i>	Insert code Lesson: F43

**Data transmission program.**

We would like to develop a program, which transmits data from Module Z3/EV to a serial peripheral connected to it.

Keeping in mind of what was seen in the theoretical part of the lesson, this program must operate according to the flow diagram of fig. F43.7.

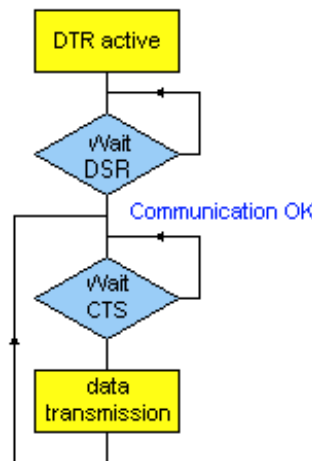


fig. F43.7

From an analysis the flow diagram it can be noted:

- module Z3 sets active the signal DTR to indicate the remote unit that it wants to activate a communication,
- the remote unit reads this signal by means the line DSR and sets active the own line DTR, too.
- when module Z3 finds line DSR active, it considers open a communication session and proceeds,
- the remote unit signals that it's ready to receive data by means of the line RTS,
- module Z3 waits until the line CTS (coming from the line RTS of the remote unit) is active, before transmitting each single data,
- module Z3 transmits code ASCII 01010101B,
- the program of module Z3 proceeds returning to waiting for the signal CTS to transmit the next byte.

The list of the program is shown next.

```

1          PAGE 70,166
2          ;----- PRG_TX -----
3          ; data transmission on the serial interface
4          ;parameters TX: 1200, N, 8, 1
5          ;waiting of DSR at the beginning of the connection
6          ; waiting of CTS at each byte to transmit
7 = 0800    MEM_POS    = 0800H
8 = 0080    DS_SEG     = 0080H
9 = 01F4    DELAY      = 500      ;retard in ms
10 = 000B    IDIS_STR   = 0BH      ; visual interruption.  stringe
11 = 000D    IWAIT_MS   = 0DH      ; management interruption waiting
12 = 0014    ISERIAL    = 14H      ; management interruption serial
13 = 000F    IDA        = 0FH
14
15          ;----- CODE
16          ;program charged in address 0000:MEM_POS
17 0000     CODE        SEGMENT
18                                ASSUME CS:CODE, DS:CODE
19 0000     ORG 00H
20 0000 B8 0080 START:  MOV AX,DS_SEG
21 0003 B8 0080        MOV AX,DS_SEG
22 0006 8E D8          MOV DS,AX      ;charges data segment
23 0008 BE 0042 R      MOV SI,OFFSET MW_DSR
24 000B CD 0B          INT IDIS_STR   ;display message MSG
25 000D B4 00          MOV AH,00H
26 000F B0 63          MOV AL,01100011B
27 0011 CD 14          INT ISERIAL    ; serial programing
28 0013 B4 04          MOV AH,04H
29 0015 B0 01          MOV AL,00000001B ;DTR active
30 0017 CD 14          INT ISERIAL
31 0019 B4 03 W_DSR:   MOV AH,03H
32 001B CD 14          INT ISERIAL
33 001D 80 E4 20       AND AH,00100000B ;check DSR
34 0020 74 F7          JZ W_DSR       ;loop if not
35
36 0022 BE 0053 R TLOOP: MOV SI,OFFSET MW_CTS
37 0025 CD 0B          INT IDIS_STR   ;display message MSG
38 0027 B4 03 W_CTS:   MOV AH,03H
39 0029 CD 14          INT ISERIAL
40 002B 80 E4 10       AND AH,00010000B ;check CTS
41 002E 74 F7          JZ W_CTS       ;loop if not
42
43 0030 B4 01          MOV AH,01H
44 0032 B0 55          MOV AL,01010101B
45 0034 CD 14          INT ISERIAL    ;transmission data
46 0036 BE 0064 R      MOV SI,OFFSET MS_BYTE
47 0039 CD 0B          INT IDIS_STR   ;display message MSG
48 003B B8 07D0        MOV AX,2000
49 003E CD 0D          INT IWAIT_MS
50 0040 EB E0          JMP TLOOP
51
52 0042 57 61 69 74 20 66 MW_DSR DB 'Wait for DSR ',00H
53      6F 72 20 44 53 52
54      20 20 20 20 00
55 0053 57 61 69 74 20 66 MW_CTS DB 'Wait for CTS ',00H
56      6F 72 20 43 54 53
57      20 20 20 20 00
58 0064 42 79 74 65 20 74 MS_BYTE DB 'Byte transmitted ',00H
59      72 61 73 6D 65 73
60      73 6F 20 20 00
61
62 0075          CODE        ENDS
63          END START

```

Insert this program in Module Z3/EV using the keyboard (in the case a Personal Computer is being used, use the application MODZ3 to transfer the program PRG\_TX via serial or parallel, using the adequate cables issued).

Give the command: LD\_KB

Specify the departing address: 0000:0800

Insert the codes of the program as from the list.

This experience requires the use of a remote communication unit. Next, the use of a PC as remote unit is foreseen.

Connect module Z3 to the PC by means of the adequate serial communication cable.

Start the execution on the PC of a communication program (it's possible to use the Windows terminal present in the accessories).

Sets the communication speed at 1200 baud, 8 bit, no parity, 1 stop bit.

Preset the flow control of hardware type (which uses therefore the lines DTR, DSR, CTS, RTS).

Give the command GO 0080:0000 (or RUN) of program execution.

**Q1** *What letter, transmitted by module Z3, appears on the computer ?*

**SET**

*A B*

**1 2 A**

**2 5 a**

**3 1 G**

**4 3 U**

**5 4 u**

<b>⇒ SIS1</b>	<b>Set switch S13 in the ON position</b>
<b>⇒ SIS2</b>	<b>Press INS</b>

**Q2** *It can be observed now how the transmission stops. What is the cause of this stop ?*

**SET**

*A B*

**1 5** The transmission line Tx is interrupted and the signal is not present at the corresponding test-point

**2 4** The clock for the functioning of the 8250 is not present at the corresponding input pin

**3 2** The oscillator at 1,8432 MHz doesn't work

**4 3** The chip's enabling signal doesn't arrive to the 8255 (CS2#)

**5 1** The data bus signals do not arrive correctly to the 8250.

➡ <i>SIS1</i>	Set switch S13 in the <i>OFF</i> position Set switch S11 in the <i>ON</i> position
➡ <i>SIS2</i>	Press <i>INS</i>

**Q3**      *The transmission of data continues to be absent. What is the cause of this new malfunction?*

**SET**

*A    B*

- |          |          |   |
|----------|----------|---|
| <b>1</b> | <b>4</b> | The transmission line Tx is interrupted and the signal is not present at the corresponding test-point |
| <b>2</b> | <b>5</b> | The clock for the functioning of the 8250 is not present at the corresponding input pin               |
| <b>3</b> | <b>1</b> | The oscillator at 1,8432 MHz doesn't work   |
| <b>4</b> | <b>2</b> | The chip's enabling signal doesn't arrive to the 8255 (CS2#)  |
| <b>5</b> | <b>3</b> | The data bus signals do not arrive correctly to the 8250.   |

➡ <i>SIS1</i>	Set switch S11 in the <i>OFF</i> position
---------------	---



### Data reception program.

We would like to develop a program which receives data in Module Z3/EV coming from a serial peripheral connected to it.

Keeping in mind of what we have seen on the theoretical part of the lesson, this program has to operate according to what is indicated in the flow diagram of fig. F43.8.

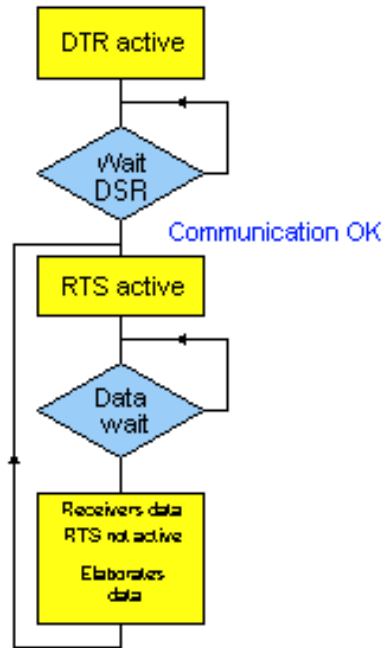


fig. F43.8

From an analysis of the flow diagram it can be noted that:

- module Z3 sets active the signal DTR to indicate the remote unit that it wants to activate a communication,
- the remote unit reads this signal by means the line DSR and sets active the own line DTR, too.
- when module Z3 finds line DSR active, it considers open a communication session and proceeds,
- the remote unit signals that it's ready to receive data by means of the line RTS,
- module Z3 waits until the line CTS (coming from the line RTS of the remote unit) is active, before transmitting each single data,
- module Z3 goes into data reception
- once it has received data the module Z3 inactivates the line of RTS to block the transmission of other data
- the program of module Z3 proceeds to the elaboration of the received data (display on display)
- the program of module Z3 returns in cycle reactivating the line RTS to request the transmission of further data.

The list of the program is shown next.

```

1          PAGE 70,166
2          ;----- PRG_RX -----
3          ; data reception from the serial interface
4          ;parameter of RX: 1200, N, 8, 1
5          ;enables DTR at the beginning of connection
6          ; enables RTS at each byte to receive
7 = 0800    MEM_POS    = 0800H
8 = 0080    DS_SEG     = 0080H
9 = 01F4    DELAY      = 500      ;delay in ms
10 = 0009    IDIS_BYTE  = 09H      ;interruption vis. byte
11 = 000B    IDIS_STR   = 0BH      ;interruption visual. stringe
12 = 000D    IWAIT_MS   = 0DH      ;interruption management wait
13 = 0014    ISERIAL    = 14H      ;interruption management serial
14 = 000F    IDA        = 0FH
15
16          ;----- CODE
17          ;program charged in address 0000:MEM_POS
18 0000      CODE      SEGMENT
19          ASSUME CS:CODE, DS:CODE
20 0000      ORG 00H
21 0000 B8 0080 START:  MOV AX,DS_SEG
22 0003 B8 0080        MOV AX,DS_SEG
23 0006 8E D8          MOV DS,AX      ;charges data segment
24 0008 B4 00          MOV AH,00H
25 000A B0 63          MOV AL,01100011B
26 000C CD 14          INT ISERIAL    ;programing serial
27 000E B4 04          MOV AH,04H
28 0010 B0 01          MOV AL,00000001B ;DTR active
29 0012 CD 14          INT ISERIAL
30
31 0014 BE 0052 R      MOV SI,OFFSET MW_DSR
32 0017 CD 0B          INT IDIS_STR    ;display message MSG
33 0019 B4 03 W_DSR:   MOV AH,03H
34 001B CD 14          INT ISERIAL
35 001D 80 E4 20       AND AH,00100000B ;check DSR
36 0020 74 F7          JZ W_DSR        ;loop if not
37
38 0022 BB 0000        MOV BX,0000H    ;counter
39 0025 BE 0063 R      MOV SI,OFFSET M_BYTE
40 0028 CD 0B          INT IDIS_STR    ;display message
41
42 002A B4 04 TLOOP:   MOV AH,04H
43 002C B0 03          MOV AL,00000011B ;DTR=ON, RTS=ON
44 002E CD 14          INT ISERIAL
45 0030 B4 02          MOV AH,02H
46 0032 CD 14          INT ISERIAL    ;reads data
47 0034 50             PUSH AX
48 0035 B4 04          MOV AH,04H
49 0037 B0 03          MOV AL,00000011B ;DTR=OFF, RTS=ON
50
51 0039 58             POP AX
52 003A B1 0E          MOV CL,14
53 003C CD 09          INT IDIS_BYTE   ;displays data
54 003E 8A C7          MOV AL,BH
55 0040 B1 05          MOV CL,5
56 0042 CD 09          INT IDIS_BYTE   ; displays MSB address
57 0044 8A C3          MOV AL,BL
58 0046 B1 07          MOV CL,7
59 0048 CD 09          INT IDIS_BYTE   ; displays LSB address
60
61 004A 43             INC BX
62 004B B8 00C8        MOV AX,200
63 004E CD 0D          INT IWAIT_MS
64 0050 EB D8          JMP TLOOP
65
66 0052 57 61 69 74 20 66 MW_DSR DB 'Wait for DSR ',00H
67 6F 72 20 44 53 52
68 20 20 20 20 00
69 0063 4E 75 6D 2E 20 78 M_BYTE DB 'Num. xxxx B. xx',00H
70 78 78 78 20 20 42
71 2E 20 78 78 00
72
73 0074      CODE      ENDS
74          END START

```

The program allows visualizing on the display the hexadecimal code of the received byte (on the right part) and the progressive count of the received byte (on the central part).

Insert this program in Module Z3/EV using the keyboard (in case in which a Personal Computer is used, use the application MODZ3 to transfer the program PRG\_RX via serial or parallel, using the adequate cables issued).

Give the command: LD\_KB

Specify the departing address: 0000:0800

Insert the codes of the program as listed

This experience requires the use of a communication remote unit. Next, the use of a PC as remote unit is analyzed.

Connect the module Z3 to the PC by means of the adequate serial communication cable.

Start the execution on the PC of a communication program (It's possible to use the Windows terminal present in the accessories).

Sets the communication speed at 1200 baud, 8 bit, no parity, 1 stop bit.

Preset then the flow control of the hardware type (which uses therefore the lines DTR, DSR, CTS, RTS).

Give the command GO 0080:0000 (o RUN) of program execution.

Press on the PC's keyboard the key corresponding to the letter 'A'.

**Q4**      *What hexadecimal code, received by module Z3, appears on the display of the module (on the right part) ?*

**SET**

<i>A</i>	<i>B</i>	
<b>1</b>	<b>3</b>	00
<b>2</b>	<b>5</b>	FF
<b>3</b>	<b>4</b>	32
<b>4</b>	<b>2</b>	68
<b>5</b>	<b>1</b>	41

➡ <i>SIS1</i>	Set switch S9 in the <i>ON</i> position
➡ <i>SIS2</i>	Press <i>INS</i>

**Q5** *It can be observed now how, by pressing any key of the PC's keyboard, the display of module Z3 is not updated.  
What is the cause of this malfunction ?*

**SET**

<i>A</i>	<i>B</i>	
----------	----------	--

- |          |          |   |
|----------|----------|---|
| <b>1</b> | <b>4</b> | The signal DTR is not active anymore and the transmission is interrupted                |
| <b>2</b> | <b>1</b> | The clock for the functioning of the 8250 is not present at the corresponding input pin |
| <b>3</b> | <b>5</b> | The reception signal RXD is not present at the input of the 8250                        |
| <b>4</b> | <b>2</b> | The chip's enabling signal of the 8250 (CS2 #) doesn't arrive                           |
| <b>5</b> | <b>3</b> | The signals of the data bus don't arrive correctly to the 8250.                         |

➡ <i>SIS1</i>	Set switch S9 in the <i>OFF</i> position
---------------	--

## LESSON F44: DIGITAL/ANALOG CONVERSION

### OBJECTIVES

- Use and principles of the digital/analog conversion
- Study of the converter DAC0800
- Application of the converter DAC0800 in module Z3
- Development of application programs and exercises.

### MATERIALS

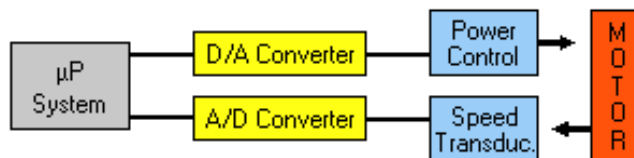
- Base unit for the IPES system (power supply mod.PS1-PSU/EV, module holder mod.MU/EV, individual management unit mod.SIS1/SIS2/SIS3)
- Experimentation module mod. Z3/EV
- oscilloscope.

### F44.1 USE of the DIGITAL/ANALOG CONVERSION

The microprocessors are used nowadays, in many applications; not only in systems digital, but also in analog systems.

Then, there is the problem of the transformation of analog signals in digital and viceversa.

Let's consider the case of figure F44.1, where a microprocessor controls the rotation speed of a motor:



*fig. F44.1*

Let's note the presence:

- An Analog/Digital converter to read the turning speed of the motor
- a Digital/Analog converter to furnish the command signal to the power electronics of the motor.

There are different types of converters with different performances referring to:

- conversion time
- resolution
- input and output fields
- etc.

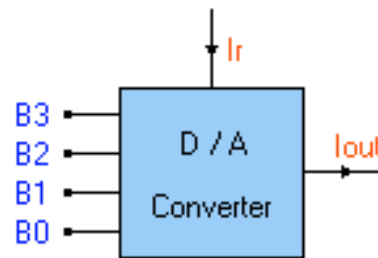
In this lesson we will study the converter DAC0800 used in the Module Z3.

## F44.2 PRINCIPLES of the DIGITAL/ANALOG CONVERSION

A Digital/Analog converter transforms a digital information (the logic state of one or more input bit) in an analog signal, which is normally a current.

Let's take into consideration the example of Figure F44.2.

- $I_r$  is the reference current in input
- B0-B3 are the digital inputs of the converter
- $I_{out}$  is the output current



*fig. F44.2*

The output current  $I_{out}$  is given by a weighted addition of the input current  $I_r$ .

The weights are obtained by means of resistive nets and the addition is obtained by means of switches CMOS which become active when the corresponding input is at the logic level "1".

The weight net is called Ladder Network, and the precision of the resistance and of the reference current  $I_r$ , determine the precision of the Analog/Digital converter.

## F44.3 CONVERTER DAC0800: characteristics

The DAC-0800 is a monolithic digital-analog converter, of 8 bit, at high conversion speed (100 ns) and with current output.

It is provided with complementary current output, which allow obtaining differential output voltages with a simple load resistance.

The inputs with immunity to noise accept TTL levels with the threshold logic pin  $V_{LC}$  set to ground.

By changing the voltage at pin  $V_{LC}$  it's possible to interface the other logic families.

The performances and characteristics of the device remain unchanged over the whole range of admitted supplies.

The dispersion of power is of only 33 mW with supply of  $\pm 5V$  and is independent from the state of the logic inputs.

The table containing the characteristics is shown next:

Characteristics	Values
Quick ordering of the output current	100 ns
Maximum error along the entire scale	$\pm 1$ LSB
Non linearity with the temperature	$\pm 0.1\%$
Current of maximum derivation	$\pm 10$ ppm/ $^{\circ}\text{C}$
High output voltage	-10V a +18V
Complementary current outputs	
Direct interface with TTL, CMOS, PMOS, ..	
High supply fields	$\pm 4.5\text{V}$ a $\pm 18\text{V}$
Low power consumption	33 mW a $\pm 5\text{V}$
Low cost	

#### F44.4 CONVERTER DAC0800: pin-out

The pin-out of the converter DAC-0800 is shown in figure F44.3 .



fig. F44.3

The signals used are:

- $V_{LC}$ : (Threshold Control). Signal for the adaptation to the logic families ( $V_{LC}=0\text{V}$  for the TTL).
- $V+$ : Positive supply voltage
- $V-$ : Negative supply voltage
- COM: Compensation
- $V_{REF+}$ : Positive reference voltage.
- $V_{REF-}$ : Negative reference voltage (connected to 0V for the unipolar functioning).
- $I_{OUT+}$ : Output current entering the converter.
- $I_{OUT-}$ : Output current exiting the converter.
- B1-B8: Digital inputs.

### F44.5 CONNECTION DIAGRAM of MODULE Z3/EV

The electrical diagram of the circuit corresponding to the digital/analog conversion, at the interior of module Z3/EV, is shown in Fig. F44.4 .

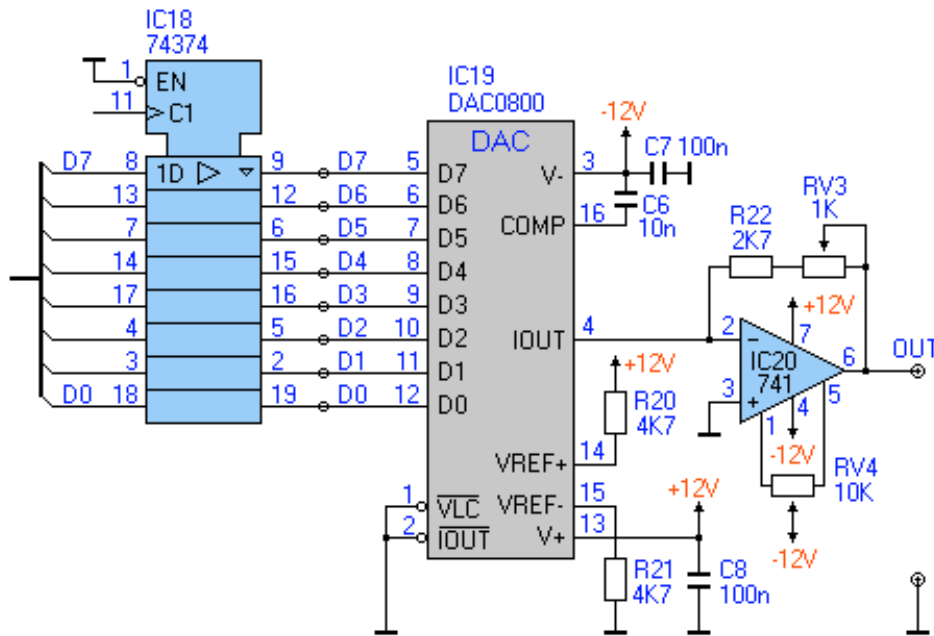


fig. F44.4

From the diagram it can be observed:

- The latch IC18 (74374) provides the 8 digital input bit to the D/A converter.
- The latch is enabled in address 0300H.
- The DAC 0800 is supplied with voltages +12V, -12V.
- The resistance R20 determines the value of the reference current  $I_r$ :  

$$I_r = 12 / R20 = 12 / 4K7 = 2 \text{ mA}$$
- The operational amplifier IC20 transforms the current of the converter in a voltage signal:  

$$V_{out} = (R22 + Rv3) * I_r$$
- The potentiometer RV3 allows regulating the maximum value of the output voltage (8 V).
- The potentiometer RV4 allows regulating the offset of the output signal.

The **interruption software INT 0FH** allows controlling in a simple mode the D/A converter. It receives the data at the register AL and sends it directly to port 0300H of the converter.



**F44.6 EXERCISES and SUMMARY QUESTIONNAIRE**

➡ <b>Z3</b>	
➡ <b>SIS1</b>	<b>Set all switches in the <i>OFF</i> position</b>
➡ <b>SIS2</b>	<b>Insert code Lesson: F44</b>

We would like to develop a program for the Module Z3/EV that uses the D/A converter to generate a waveform with saw teeth shape.

The list of the program is shown next

```

1          PAGE 70,166
2          ;----- PRG_DA -----
3          ; generating a waveform with saw teeth shape in
4          ;output of the Digital/Analog converter
5 = 0800    MEM_POS    = 0800H
6 = 000F    IDA_WRITE  = 0FH      ;interruption management conv. D/A
7
8          ;----- CODE -----
9          ;program charged in address 0000:MEM_POS
10 0000     CODE      SEGMENT
11          ASSUME CS:CODE, DS:CODE
12 0000     ORG      0H
13 0000 B0 FF  START:  MOV  AL,0FFH
14 0002 FE C0  TLOOP:  INC  AL
15 0004 CD 0F      INT  IDA_WRITE      ;out to D/A
16 0006 EB FA      JMP  TLOOP
17 0008     CODE      ENDS
18          END  START

```

Insert this program in Module Z3/EV using the keyboard (in the case a Personal Computer is being used, use the application MODZ3 to transfer the program PRG\_DA via serial or parallel, using the adequate cables issued).

Give the command: LD\_KB

Specify the departing address: 0000:0800

Insert the codes of the program as from the list.



Give the command GO 0080:0000 (o RUN) of program execution.

**Q1** *What is the variation field of the saw teeth like signal generated in output at the D/A converter?*

**SET**

*A B*

- |          |          |                |
|----------|----------|----------------|
| <b>1</b> | <b>3</b> | 0 ÷ 8 V        |
| <b>2</b> | <b>4</b> | 0 ÷ 5 V        |
| <b>3</b> | <b>1</b> | -8 ÷ +8 V      |
| <b>4</b> | <b>5</b> | -5 ÷ +5 V      |
| <b>5</b> | <b>2</b> | TTL compatible |

 <b>SIS1</b>	<b>Set switch S17 in the <i>ON</i> position</b>
 <b>SIS2</b>	<b>Press <i>INS</i></b>

**Q2**      *The waveform in output of the D/A converter has changed.  
What is the change that has been introduced ?*

**SET**

*A    B*

- |          |          |                               |
|----------|----------|-------------------------------|
| <b>1</b> | <b>5</b> | The amplitude has diminished  |
| <b>2</b> | <b>4</b> | The amplitude has augmented   |
| <b>3</b> | <b>2</b> | The frequency has diminished  |
| <b>4</b> | <b>3</b> | The frequency has augmented   |
| <b>5</b> | <b>1</b> | The waveform type has changed |

**Q3**      *What is the cause of this change ?*

**SET**

*A    B*

- |          |          |  |
|----------|----------|--|
| <b>1</b> | <b>5</b> | The waveform generation algorithm has changed in the program   |
| <b>2</b> | <b>1</b> | The amplification of the operational amplifier IC20 which transforms the current signal in voltage has changed |
| <b>3</b> | <b>4</b> | The signal on bit D7 in input of the D/A converter is missing  |
| <b>4</b> | <b>3</b> | The signal on bit D6 in input of the D/A converter is missing  |
| <b>5</b> | <b>2</b> | The reference voltage VREF+ is halved.   |

 <b>SIS1</b>	<b>Set switch S17 in the <i>OFF</i> position</b>
---	--

**Q4** *Of what type is the signal in output of the D/A converter DAC0800 ?*

**SET**

<i>A</i>	<i>B</i>	
1	5	Current
2	4	Voltage
3	2	Frequency
4	3	TTL
5	1	CMOS

**Q5** *What input signal of the DAC converter 0800 is used to adapt the converter to the different logic families ?*

**SET**

<i>A</i>	<i>B</i>	
1	2	$V_{REF+}$
2	1	COM
3	4	B8
4	5	$V_{LC}$
5	3	$I_{OUT+}$ .

**Q6** *What is the value of the reference current in input at pin  $V_{REF+}$  of the DAC converter 0800 in module Z3/EV ?*

**SET**

<i>A</i>	<i>B</i>	
1	4	1 A
2	5	0.5 A
3	1	50 mA
4	2	2 mA
5	3	1 mA.

## LESSON F45: CONVERSION ANALOG/DIGITAL

### OBJECTIVES

- Study of the principles of the analog/digital conversion
- The converter ADC0804
- The pin-out of the converter
- The analog section
- The digital section and the timings
- The connection at the interior of Module Z3/EV
- Development of application programs and exercises

### MATERIALS

- Base unit for the IPES system (power supply mod.PS1-PSU/EV, module holder mod.MU/EV, individual management unit mod.SIS1/SIS2/SIS3)
- Experimentation module mod. Z3/EV

### F45.1 THE PRINCIPLES of the ANALOG/DIGITAL CONVERSION

The analog/digital conversion transforms an analog signal (normally a voltage signal) in a digital information of the binary type composed of a certain number of bit '0' or '1'.

There are different techniques to obtain this type of conversion. One of the mostly used is known under the name of Successive Approximations Conversion, shown in figure F45.1 .

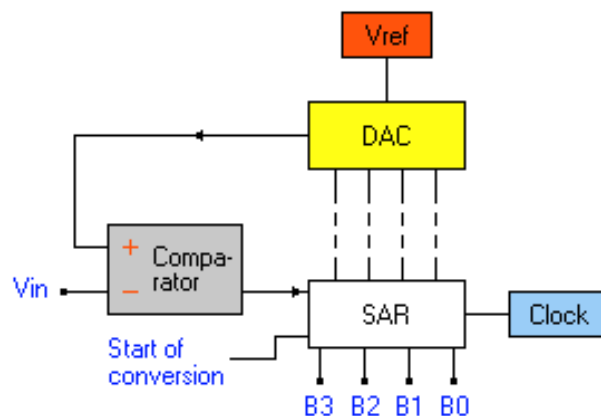


fig. F45.1

This technique uses a register Successive Approximations Register SAR (Successive Approximation Register) to obtain the sequence of states B0÷B3.

This sequence is converted in an analog signal by the D.A.C. converter, This signal is then compared with the input signal, and if the two signals are equal, the register SAR is stopped.

In this condition the digital information furnished by B0÷B3 is proportional to the analog value of the input signal and the conversion is finished.

### F45.2 THE CONVERTER ADC0804

The ADC0804 is an analog/digital converter of 8 bit that uses the technique of the successive approximations.

This converter has been projected to allow the direct connection to the bus of the microprocessors, thanks to the internal presence of a three-state latch.

The converter behaves with respect the microprocessor, as a normal input port, without requiring interface logic devices.

The presence of differential inputs for the analog voltage provide an elevated rejection rate to the signals of common mode.

The table with the characteristics is shown next:

Characteristics	Values
Resolution	8 bit
Conversion time	100 $\mu$ s
Compatibility with the microprocessors. Access time:	135 ns
Analog differential voltage inputs	
Logic inputs and outputs compatible TTL and CMOS	
Chip internal clock generator	
Supply	5 V
Input analog voltage field	0-5 V
Socket of 20 pin standard	

### F45.3 CONVERTER ADC0804: pin-out

The pin-out of the converter ADC0804 is shown in figure F45.2 .



fig. F45.2

The signals used are:

V <sub>CC</sub> :	Supply positive.
GND	Analog ground
CLK IN	Input clock
CLK R	Resistance and capacity for the internal oscillator
Vin(+)	Positive differential input
Vin(-)	Negative differential input
VREF/2	Reference voltage
DB0÷7	Data output (8 bit)
INTR	Interruption Signal (end of conversion)
CS	Signal selection of the device
WR	Writing signal
RD	Reading signal
D GND	Digital ground.

#### F45.4 CONVERTER ADC0804: the analog section

##### Clock

The clock for the functioning of the converter ADC0804 can be obtained from the clock of the CPU and then sent to the pin CLK-IN of the converter.

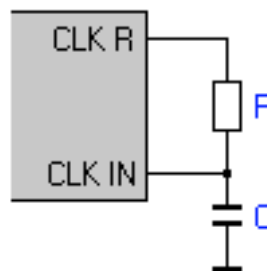
Otherwise, we can use the internal clock of the converter, adding an external RC, as in Fig. F45.3 .

Using an external resistance of about 10 Kohm, the frequency can be obtained with the formula:

$$F = 1 / (1.1 \cdot R \cdot C)$$

The variation field of this frequency is: 100 ÷ 1460 kHz.

The normal value is of: 640 KHz.



*fig. F45.3*

##### Analog inputs

The converter has an input of analog differential voltage.

The input VIN- can be used to subtract automatically a fixed voltage value from the reading input.

Normally, when the input field is between 0 and 5 V, the connections to be done are those shown in figure F45.4 .

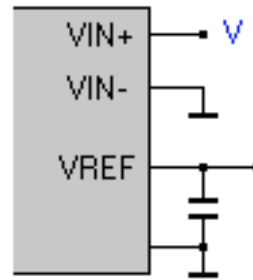


fig. F45.4

The value of the reference voltage can be  $\frac{1}{2}$  of the voltage applied to the supply pin Vcc, otherwise it can be equal to the value of the voltage applied externally to pin VREF.

In the figure, the internal reference is used. To obtain this, it's enough to connect a ground to pin VREF with a capacitor of  $0.1\mu\text{F}$ .

### F45.5 CONVERTER ADC0804: the digital section and the timings

The converter ADC0804 has been projected to be able to being directly interfaced to the bus of the microprocessors.

The signals used for this interfacing are:

- CS: signal selection of the chip.
- WR: writing signal (starts a new conversion)
- INTR: interruption signal (signals when the converter is converting and when the data is available in output)
- RD: reading signal (it's used to put the result of the conversion on the bus, in a way to allow its reading from the microprocessor).

The timings for the command operations of the conversion, the waiting of the end of conversion and the data reading are shown in Fig. F45.5 .

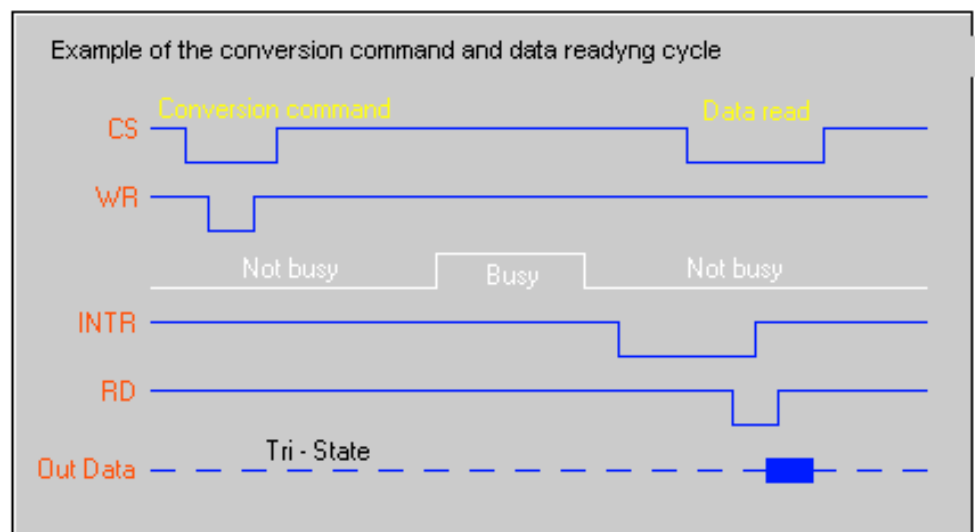


fig. F45.5

See how:

- a writing operation (WR#) activates a new conversion
- during the conversion the line INTR is at high level
- when the line INTR returns to be low, the conversion is finished and the converter is free
- a reading operation (RD#) allows then to read the data.

#### F45.6 CONNECTION DIAGRAM of MODULE Z3/EV

The electrical diagram of the circuit corresponding to the analog/digital conversion, at the interior of module Z3/EV, is shown in Fig. F45.6 .

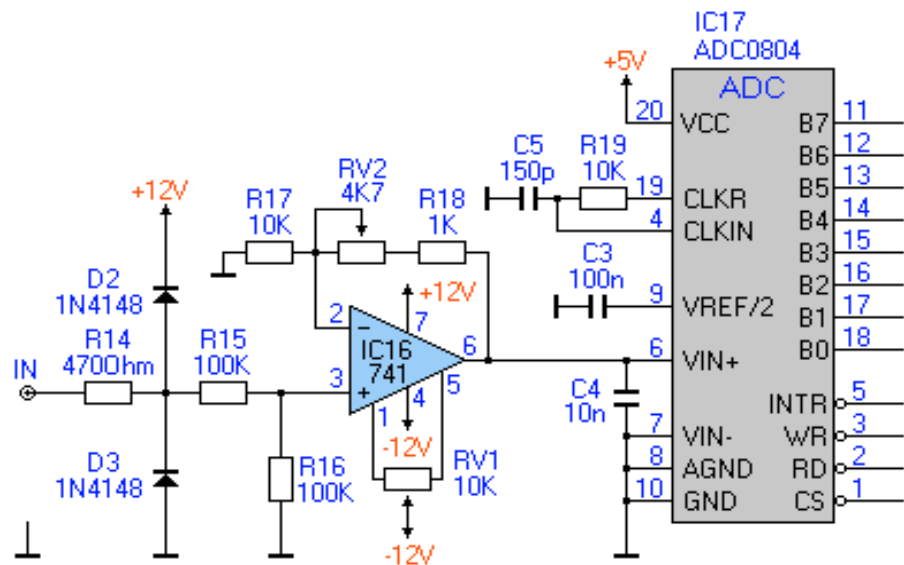


fig. F45.6

From the diagram it can be observed:

- The signal in input of module Z3/EV is previewed in the field of 0÷8 Volt.
- Since the converter ADC0804 works with a variation field of 0÷5 V, the operational amplifier IC16 ( $\mu$ A741) is used as an amplifier for voltage reduction.
- The potentiometer RV2 allows regulating the maximum value of the output voltage at the amplifier (5 V).
- The potentiometer RV1 allows regulating the offset of the output signal.
- The ADC0804 is supplied with a single voltage of +5V.
- The resistance R10 and the capacitor C5 compose the external net for the internal oscillator.
- The internal reference is used, once connected VREF to ground by means of a capacitor.
- The RD# and WR# lines are connected to the analog lines of the microprocessor.



- The line CS# arrives from the decoding of the I/O addresses.
- The line INTR is sent to the input INT0 of the microprocessor, being able to use this way, this interruption in the reading operations of the converter.
- The line INTR is also sent to the IC21 (used even to read the keyboard). Thus, it's possible to read the state of this line, making input operations to address 032EH and making a test on the most significant bit (D7). The interruption software INT 0EH, which allows reading from the A/D converter, uses this modality.

For the command and reading of the A/D converter it's possible to use the **interruption software INT 0EH**.

It operates in the following mode:

- sends the command of conversion starting (WR#) to the converter
- waits for the signal of end of conversion (line INTR)
- reads the result of the conversion
- returns the result of the conversion into register AL.

## F45.7 EXERCISES and SUMMARY QUESTIONNAIRE

➡ Z3	
➡ SIS1	Set all switches in the <i>OFF</i> position
➡ SIS2	Insert code Lesson: F45

We would like to develop a program for the Module Z3/EV that reads from the A/D converter and visualizes the result of the conversion in hexadecimal form on the display.

The list of the program is shown next.

```

1          PAGE 70,166
2          ;----- PRG AD -----
3          ;reading from the Analog/Digital converter
4          ;and displays the result result
5 = 0800    MEM_POS    = 0800H
6 = 0080    DS_SEG     = 0080H
7 = 000E    IAD_READ   = 0EH      ;interruption reading conv. A/D
8 = 0009    IDIS_BYTE  = 09H      ;interruption visual. byte
9 = 000B    IDIS_STR   = 0BH      ;interruption visual. stringe
10 = 000D    IWAIT_MS  = 0DH      ;interruption wait
11          ;----- CODE -----
12          ;program charged in address 0000:MEM_POS
13 0000      CODE      SEGMENT
14                      ASSUME CS:CODE, DS:CODE
15 0000      ORG 0H
16 0000 B8 0080 START:  MOV AX,DS_SEG
17 0003 8E D8          MOV DS,AX      ;charges data segment
18 0005 BE 0017 R      MOV SI,OFFSET MSG
19 0008 CD 0B          INT IDIS_STR   ;display message MSG
20 000A CD 0E TLOOP:   INT IAD_READ   ;reads A/D
21 000C B1 0D          MOV CL,13
22 000E CD 09          INT IDIS_BYTE  ;visualizes code key
23 0010 B8 00C8        MOV AX,200
24 0013 CD 0D          INT IWAIT_MS
25 0015 EB F3          JMP TLOOP
26
27 0017 20 41 2F 44 20 76 MSG      DB ' A/D value : xx ',00H
28      61 6C 75 65 20 3A
29      20 78 78 20 00
30
31 0028      CODE      ENDS
32          END START

```

Insert this program in Module Z3/EV using the keyboard (in case in which a Personal Computer is used, use the application MODZ3 to transfer the program PRG\_AD via serial or parallel, using the adequate cables issued).

Give the command: LD\_KB

Specify the departing address: 0000:0800



Insert the codes of the program: B8, 80, 00, ....., 20, 00.

Give the command GO 0080:0000 (or RUN) of program execution.

- Q1** *Set, by means of an external supply source, a signal of 6 Volt to the input of the A/D.  
What is the hexadecimal value visualized on the display (indicate the closest value) ?*

**SET**

<i>A</i>	<i>B</i>	
1	4	20
2	3	7F
3	5	BF
4	2	CF
5	1	DF

 <b>SIS1</b>	<b>Set switch S18 in the <i>ON</i> position</b>
 <b>SIS2</b>	<b>Press <i>INS</i></b>

- Q2** *The value indicated on the display has changed.  
What is the cause of this change ?*

**SET**

<i>A</i>	<i>B</i>	
1	5	The ADC converter doesn't work correctly
2	4	The signal CS# does not arrive to the converter
3	2	The data bus is not connected
4	3	The reference voltage VREF has changed
5	1	The input signal does not arrive to the converter ADC0804

 <b>SIS1</b>	<b>Set switch S18 in the <i>OFF</i> position</b>
---	--

- Q3** *Of what type is the signal at the input of converter ADC0804 ?*

**SET**

<i>A</i>	<i>B</i>	
1	2	Current
2	1	Voltage
3	5	Frequency
4	3	TTL
5	4	CMOS

**Q4**      *What input signal of the converter DAC 0800 is used to adapt the converter to the different logic families ?*

**SET**

<i>A</i>	<i>B</i>	
1	5	$V_{REF+}$
2	1	COM
3	4	B8
4	3	$V_{LC}$
5	2	$I_{OUT+}$ .

**Q5**      *What is the value of the clock frequency of the internal oscillator of the converter ADC0804, in the case in which an external net is used, with  $R=15k\Omega$  e  $C=150pF$  (indicate the closest calculated value) ?*

**SET**

<i>A</i>	<i>B</i>	
1	4	400 kHz
2	5	500 KHz
3	1	700 KHz
4	2	800 KHz
5	3	1 MHz.

## A. MONITOR COMMANDS of MODULE Z3/EV

### A. 1 INTRODUCTION

The MONITOR is the program found at the interior of the system's EPROM, which provides the management of the system and allows the user to work with it. Its fundamental functions are:

- display and modification of memory and registers
- continuous execution, step, with program's breakpoints
- charging of programs from keyboard and from PC.

The interaction with the user is done by means of the keyboard and the display.

### A. 2. KEYBOARD

The keyboard looks like shown in figure:

CHG/RET	→	←	RESET
GEN C	BR D	CB E	LAST F
RUN 8	GO 9	SS A	FIRST B
LD-KB 4	LD-PAR 5	LD-SER 6	DEC (-) 7
MEM 0	REG 1	SEG 2	INC (+) 3

It is basically divided in two parts:

- The section of 4 top keys.
- The lower section with 16 hexadecimal keys with numeric function and command

**NOTE:**

The microprocessor 80386 addresses a high quantity of memory. The devices (RAM and EPROM) present in the system occupy not all of this memory.

**If trying to access, during the use of the Monitor commands, to non occupied memory cells from the RAM or from the EPROM, the system blocks, since the microprocessor sets itself to wait for the signal READY from the external memory (which is obviously not there).**

**In these cases, a system reset has to be operated in order to return to the command mode.**

### Section of the top 4 keys.

In This section there are present:

A rectangular button with a light gray background and a black border. The word "RESET" is written in black, uppercase letters in the center.

This key is physically connected to the reset line of the microprocessor. It's always active and provokes the reset of the micro and the starting of the program Monitor in the EPROM of the system.

A rectangular button with a light gray background and a black border. The text "CHG/RE" is written in black, uppercase letters in the center.

This key has a double function. The function CHG (Change) allows entering the *modification session* of an eventual value present on the display (contents of a register, memory site, ..) The function RET (Return) allows finishing the *modification session*. The *modification session* is made evident from the presence of the cursor on the display.



The **ARROW** keys allow moving the cursor during the *modification session*.

### Section 16 Hexadecimal/Command Keys.

The 16 keys in this section have a double function.

**Numeric function.** Correspond to the 16 hexadecimal numbers 0-F during the *modification sessions* (the modification session can be entered with key CHG/RET).

**Command function.** They allow giving the commands to the Monitor when the system is not in modification session (the modification session can be exited with key CHG/RET).

### A. 3 COMMANDS OF THE MONITOR

The commands of the Monitor are shown next and make reference to the description of the keyboard's keys, which corresponds to each one of the commands.

#### MEM

This command allows examining the contents of the memory.

The display is done one byte at the time.

Once the key has been pressed, the address of the memory cell to be visualized in the segment form, is requested: address (the system is set directly in modification session):

A	d	d	r	.			s	s	s	s	:	i	i	i	i
---	---	---	---	---	--	--	---	---	---	---	---	---	---	---	---

Once the address of interest has been inserted, the key CHG/RET has to be pressed in order to end the modification session. On the display appear at this time the specified address and the corresponding data, in the form:

s	s	s	s	:	i	i	i	i		a	x		h	x	x
---	---	---	---	---	---	---	---	---	--	---	---	--	---	---	---

On the left appears the address in the segment form:address.

the character, which appears after the letter 'a', represents the data in ASCII code.

The number, which appears after the letter 'h', represents the data in hexadecimal form.

The possible commands at this time are the following:

- To pass to the successive cell: press INC(+)
- To pass to the preceding cell: press DEC(-)
- To pass to the first RAM user cell: press FIRST
- To pass to the last RAM user cell: press LAST
- To modify the displayed data: press CHG/RET
- To end the modification: press CHG/RET.

#### REG

This command allows visualizing/modifying the contents of all the registers of the microprocessor:

EAX, ECX, EDX, ESI, EDI, EBP, ESP, EIP, EFLAGS

After having pressed the key, the first register appears on the display:

R	e	g	.	E	A	X		x	x	x	x	x	x	x	x
---	---	---	---	---	---	---	--	---	---	---	---	---	---	---	---

The possible commands at this time are the following:

- To pass to the successive register: press INC(+)
- To pass to the preceding register: press DEC(-)
- To pass to the first register: press FIRST
- To pass to the last register: press LAST
- To modify the value of the register: press CHG/RET
- To end the modification of the register: press CHG/RET

**SEG**

This command allows visualizing/modifying the contents of all the segment registers of the microprocessor:  
CS, SS, DS, ES, FS, GS

After having pressed the key, the first segment register appears on the display:

S	e	g	R	e	g	.	C	S	:			x	x	x	x
---	---	---	---	---	---	---	---	---	---	--	--	---	---	---	---

The possible commands at this time are the following:

- To pass to the successive register: press INC(+)
- To pass to the preceding register: press DEC(-)
- To pass to the first register: press FIRST
- To pass to the last register: press LAST
- To modify the value of the register: press CHG/RET
- To end the modification of the register: press CHG/RET

**LD\_KB**

This command allows charging a program in the memory emitting the instruction codes by means of the keyboard.

Once the key has been pressed, the departing address of the program in the segment form is requested:address (the system is set directly in modification session):

A	d	d	r	.			s	s	s	s	:	i	i	i	i
---	---	---	---	---	--	--	---	---	---	---	---	---	---	---	---

Once the address of interest has been inserted, the key CHG/RET has to be pressed in order to end the modification session. On the display appear at this time the specified address and the corresponding data, in the form:

s	s	s	s	:	i	i	i	i		a	x		h	x	x
---	---	---	---	---	---	---	---	---	--	---	---	--	---	---	---

The modification session is entered with the cursor in the first position of the hexadecimal data to insert in the indicated memory site.

Once the data has been inserted, the key CHG/RET is pressed which memorizes the data, increments the memory site, and gets ready for new data insertion.

The insertion operations are terminated with the key RESET.



**LD\_PAR**

This command allows charging a program from the Personal Computer, by means of the parallel interface.

Once pressed the key, the system is set to listening of the parallel interface and memorizes all the bytes, which arrive to it, starting from the memory address 0000:0800H.

Once the transfer is finished, the system visualizes the number of bytes received and is set to the command mode.

See chapter 6 referring to the Communications with Personal Computer for the operative details.

**LD\_SER**

This command allows charging a program from the Personal Computer, by means of the serial interface.

Once pressed the key, the system is set to listening of the serial interface and memorizes all the bytes, which arrive to it, starting from the memory address 0000:0800H.

Once the transfer is finished, the system visualizes the number of bytes received and is set to the command mode.

See chapter 6 referring to the Communications with Personal Computer for the operative details.

**RUN**

This command starts a program considering the memory address 0000:0800H as the departing address.

It is useful for quickly starting the programs transferred from PC, which are positioned automatically starting from this address.

**GO**

This command allows starting the execution of a program specifying the departing address.

The departing address is required to be introduced in the form CS:IP (the system is set directly into modification session) :

G	O	f	r	o	m		s	s	s	s	:	i	i	i	i
---	---	---	---	---	---	--	---	---	---	---	---	---	---	---	---

Once the program's departing address has been inserted, the key CHG/RET has to be pressed. A this time the execution of the program is launched starting from the specified address.

**SS**

This command allows following a step of the active user program, starting from the present value contained in the user registers CS and IP. Pressed consecutively, it allows following a program step by step.

After each step of the program, it stops, displaying the memory to which it has arrived:

S	t	o	p	A	t		s	s	s	s	:	i	i	i	i
---	---	---	---	---	---	--	---	---	---	---	---	---	---	---	---

it's possible at this time, examining the contents of the registers and of the memory.

**BR**

This command allows visualizing and modifying the value of the breakpoints inserted in the program (a maximum of 5 breakpoints is foreseen).

Once pressed the key, the address of the first break points is displayed in the segment form:address:

B	r		n	.	x		s	s	s	s	:	i	i	i	i
---	---	--	---	---	---	--	---	---	---	---	---	---	---	---	---

The possible commands at this time are the following:

- To pass to the successive breakpoint: press INC(+)
- To pass to the preceding breakpoint: press DEC(-)
- To pass to the first breakpoint: press FIRST
- To pass to the last breakpoint: press LAST
- To modify the value of the breakpoint: press CHG/RET
- To cancel the breakpoint: press CB

**CB**

This command allows canceling the breakpoint presently displayed (a breakpoint is considered cancelled, and therefore not active, when it's positioned in address FFFF:FFFF).

**INC(+)**

Increments the values of the memory address, of the register, of the register segment, or of the presently displayed breakpoint.

**DEC(-)**

Decrements the values of the memory address, of the register, of the register segment, or of the presently displayed breakpoint.

**FIRST**

Positions in the first memory address, register, register segment, or breakpoint.

**LAST**

Positions in the last memory address, register, register segment, or breakpoint.

**GEN**

Key of general use, left free.

## B. RESOURCES OF THE MONITOR

### B.1 INTRODUCTION

The Monitor of the system contains at its interior the management software of the different peripherals of the 32 Bit Microprocessor Trainer.

*This software is rendered available in the simplest way possible, by means of the interruption software of the microprocessor.*

In this chapter the different interruptions are described; their functions and the required parameters.

In all the examples of this manual, the interruption software will be employed for the management of the system's peripherals.

### B.2 LIST OF THE INTERRUPTIONS SOFTWARE

The list of the interruptions software of the 32 Bit Microprocessor Trainer Mod. Z3/EV is the following:

Interruption Number	Routine Name	General Description
INT 00H	IMONITOR	Division by 0. Return to the monitor.
INT 01H	reserved	Single-step
INT 02H	Not used	Interruption non mask
INT 03H	Reserved	Breakpoints
INT 04H	Not used	Overflow
INT 05H	Not used	
INT 06H	Not used	
INT 07H	IMONITOR	End user prog. and return to the monitor
INT 08H	IKEYBOARD	Reading of a key from the keyboard
INT 09H	IDIS_BYTE	Sending of a hexadecimal byte to the display
INT 0AH	IDIS_CHAR	Sending of an ASCII character to the display
INT 0BH	IDIS_OUTS	Sending of a string to the display
INT 0CH	IDIS_CODE	Sending commands to the display
INT 0DH	IWAIT_MS	Wait in milliseconds
INT 0EH	IAD_READ	Reading from A/D converter
INT 0FH	IDA_WRITE	D/A converter Command
INT 10H	IBUZZER	Buzzer Command
INT 11H	Not used	
INT 12H	IPARAL	Management parallel interface
INT 13H	Not used	
INT 14H	ISERIAL	Management serial interface

**B. 3 DESCRIPTION OF THE INTERRUPTIONS SOFTWARE**

The different interruptions, with the corresponding parameters, are described next.

***INT 07H*****End of the program.**

This interruption terminates the execution of the program and transfers the system's control to the program Monitor.

It has to be called at the end of each program having to give up the control to the monitor at the end of its execution.

It makes appear the monitor's prompt on the display.

<b>INPUT</b>	none
<b>OUTPUT</b>	none
<b>Altered Registers</b>	none

See example PRG\_JUMP.

***INT 08H*****Reading of a key from the keyboard.**

This interruption allows reading the keys pressed on the keyboard.

It makes the scanning of the keyboard and waits until a key is pressed.

The key code (number from 0 to 18) is taken back to the register AL.

<b>INPUT</b>	none
<b>OUTPUT</b>	AL = key code pressed (0-18)
<b>Altered Registers</b>	none

See example PRG\_KB.

***INT 09H*****Sending of a byte in hexadecimal form to the display.**

This interruption allows writing a byte in hexadecimal form in any position of the display.

the position is identified by means of the contents of register CL, and the byte by means of the contents of AL.

<b>INPUT</b>	CL = position on the display (0-14) AL = byte to display
<b>OUTPUT</b>	none
<b>Altered Registers</b>	none

See example PRG\_KB.

**INT 0AH****Sending of a character ASCII to the display.**

This interruption allows writing a character ASCII in any position of the display.

the position is identified by means the contents of register CL, and the ASCII code by means of the contents of AL.

<b>INPUT</b>	CL = position on the display (0-14) AL = ASCII code of the character to display
<b>OUTPUT</b>	none
<b>Altered Registers</b>	none

**INT 0BH****Sending of a string of characters on the display.**

This interruption allows sending a string of characters on the display.

The string consists in a sequence of bytes, which correspond to different characters of the string, ending with the code 00H.

The string is identified by the contents of DS:SI.

The string is written starting from the first position of the display.

<b>INPUT</b>	DS = indicates the segment containing the string SI = indicates the beginning address of the string, at the interior of the segment DS.
<b>OUTPUT</b>	none
<b>Altered Registers</b>	none

See example PRG\_KB.

**INT 0CH****Sending Commands al Display.**

This interruption sends control commands to the display:

<b>INPUT</b>	AH = 1 Cancels the display AH = 2 Brings the cursor home AH = 3 Moves the cursor right AH = 4 Moves the cursor left AH = 5 Cursor ON AH = 6 Cursor OFF AH = 7 Brings the cursor in position (AL contains position:0-15)
<b>OUTPUT</b>	none
<b>Altered Registers</b>	none

***INT 0DH*****Wait in milliseconds.**

This interruption provokes a wait, before the return, equal to the number of milliseconds specified by the contents of register AX in input:

<b>INPUT</b>	AX = number of milliseconds
<b>OUTPUT</b>	none
<b>Altered Registers</b>	none

See example PRG\_PAR.

***INT 0EH*****Reading of the Analog/Digital converter.**

This interruption operates in the following mode:

- sends the command Beginning Conversion to the A/D converter
- waits for the signal of End Conversion
- reads the result of the conversion and returns it into AL.

<b>INPUT</b>	none
<b>OUTPUT</b>	AL = result of the conversion
<b>Altered Registers</b>	none

See example PRG\_AD.

***INT 0FH*****Command of the Digital/Analog converter.**

This interruption sends a data (byte) to the Digital/Analog converter, which transforms it automatically in an analog value.

<b>INPUT</b>	AL = data
<b>OUTPUT</b>	none
<b>Altered Registers</b>	none

See example PRG\_DA.

**INT 10H****Command of the Buzzer.**

This interruption commands the emission of sounds on the buzzer. It is possible specifying the frequency and the duration of the emitted sound.

<b>INPUT</b>	BX = duration CX = frequency
<b>OUTPUT</b>	none
<b>Altered Registers</b>	none

See example PRG\_PAR.

**INT 12H****Management of the Parallel Interface.**

This interruption manages the functioning of the parallel interface. The controller 8255 is always used in Mode 0 (Basic I/O). The available functions are determined from the contents of AH:

<b>INPUT</b>	AH = 0 Programing ports AH = 1 Sending data to port A AH = 2 Sending data to port B AH = 3 Sending data to port C AH = 4 Reading data from port A AH = 5 Reading data from port B AH = 6 Reading data from port C
<b>OUTPUT</b>	AL = data read
<b>Altered Registers</b>	none

**AH = 0 : Programing ports**

The contents of AL determines the direction (I/O) of the ports:

<b>Register AL</b>	<b>Programing</b>
Bit 0	= 0 : port A in output = 1 : port A in input
Bit 1	= 0 : port B in output = 1 : port B in input
Bit 2	= 0 : port C (C0-C3) in output = 1 : port C (C0-C3) in input

**AH = 1, 2, 3 : Sending Data to ports A, B, C**

The contents of AL determine the data to be sent to the ports.

**AH = 4, 5, 6 : Reading Data from ports A, B, C**

The contents of AL in output correspond to the data read from the port.

See example PRG\_PAR.



**INT 14H****Management of the Serial Interface.**

This interruption manages the functioning of the asynchronous serial interface RS-232.

The controller 8250 is used.

The available functions are determined from the contents of AH:

<b>INPUT</b>	AH = 0 Initialization AH = 1 Sending character AH = 2 Receiving character AH = 3 Reading State AH = 4 Control modem
--------------	---

**AH = 0 : Initialization of the communication port**

The contents of AL determines the initialization parameters:

Register AL	Programing
Bit 7,6,5	Baud rate = 000 : 1200 = 001 : 2400 = 010 : 4800 = 011 : 9600 (use the baud rate 1200 for the controllers UMB 8250)
Bit 4,3	Parity = 00 : no = 01 : odd = 10 : no = 11 : even
Bit 2	Stop Bit = 0 : 1 = 1 : 2
Bit 1,0	Word length = 10 : 7 bits = 11 : 8 bits

**AH = 1 : Transmission of a character**

The contents of AL determine the character to be sent.

Before sending character, the routine waits until eventually other preceding characters have been transmitted.

**AH = 2 : Reception of a character**

The contents of AL determine the received character.

The routine waits for the availability of a character before returning to the calling program.

**AH = 3 : Reading of the state**

The contents of AH determine the present state of the line and of the modem.

<b>Register AH</b>	<b>State of the Modem</b>
Bit 7	Received line signal detect
Bit 6	Ring indicator
Bit 5	Data set ready
Bit 4	Clear to send
Bit 3	Delta receive line signal detect
Bit 2	Trailing edge ring detector
Bit 1	Delta data set ready
Bit 0	Delta clear to send
<b>Register AL</b>	<b>State of the Line</b>
Bit 7	Time-out
Bit 6	Transmitter shift register empty
Bit 5	Transmitter holding register empty
Bit 4	Break detect
Bit 3	Framing error
Bit 2	Parity error
Bit 1	Overrun error
Bit 0	Data ready

**AH = 4 : Control of the Modem**

The contents of AH determine the present state of the Modem, which will be set.

<b>Register AL</b>	<b>State of the Modem</b>
Bit 7	0
Bit 6	0
Bit 5	0
Bit 4	Loop
Bit 3	Out2
Bit 2	Out1
Bit 1	Request To Send (RTS)
Bit 0	Data Terminal Ready (DTR)

See examples PRG\_TX and PRG\_RX.

## C. COMMUNICATION with the Personal Computer

### C.1 The MODZ3 application for Windows

The Module Z3 is issued with the application MODZ3 for Windows for the use of the Personal Computer in the development of applications. MODZ3 can be used on any PC with operative system Windows 3.x, Windows 95, Windows 98.

It is furnished with a single disc and is installed on the Personal Computer with the command:

A:>SETUP

The Setup procedure provides automatically the creation of the group of programs (**EV: 32 bit Microprocessor**) and of the launching command requested.

Together with the application there are also installed application examples. Consult the file README.TXT for details.

The main functions of the application are described next.

#### **MENU FILE: Open, New, Save, .. Files.**

These commands serve to create or to open program files of type ASCII with extension .ASM .

Once the file is open, the edit window becomes active, where it's possible inserting and modifying the source programs.

#### **MENU EDIT: Undo, Cut, Copy, Paste, Delete, Find, Find Next, Replace**

These commands allow writing the programs with all the functions of any edit program for Windows, as NotePad, WordPad, etc. .

#### **MENU COMM**

This section allows transferring the programs in machine code corresponding to the source file, which is presently active.

If, for example, the file PROVA.ASM has been opened, the file PROVA.BIN or PROVA.EXE is transferred to Module Z3/EV.

For the generation of the file PROVA.BIN it's necessary to proceed using the assembler MASM Microsoft, with the following commands:

```
MASM PROVA;  
LINK PROVA;  
EXE2BIN PROVA.EXE PROVA.BIN
```

These commands can be automated with the options of the Utility menu.

***MENU COMM: Serial transmission***

This command allows transferring a program to Module Z3/EV using the serial interface RS-232 of the Computer and of the Module.

The computer and the Module Z3 (connector J2) have to be connected together using the adequate issued cable.

The transmission is done at the speed of 1200 bit/s, without parity, with 1 stop bit and with 8 bit/character.

Before the transmission, it appears a Dialog Box where it's possible setting the requested parameters.

In particular:

- The interface used: COM1, COM2 and the other communication parameters.
- The directory where the transfer program resides
- The extension of the transfer file. It can be .BIN in case the program EXE2Bin is used to convert the file generated from the Linker, otherwise it can be .EXE in case the file generated from the Linker is directly used.
- The skipping of bytes in the file to transmit. This is important in case the files .EXE are used, which normally contain in the first 200H bytes, information which serve only in MS-DOS environment.

**MENU COMM: Parallel transmission**

This command allows transferring a program to Module Z3/EV using the parallel interface of the Computer and of the Module.

The computer and the Module Z3 (connector J2) have to be connected together using the adequate issued cable.

Before the transmission, it appears a Dialog Box where it's possible setting the requested parameters. In particular:

- The interface used: LPT1, LPT2, ..
- The directory where the transfer program resides
- The extension of the transfer file. It can be .BIN in case the program EXE2Bin is used to convert the file generated from the Linker, otherwise it can be .EXE in case the file generated from the Linker is directly used.
- The skipping of bytes in the file to transmit. This is important in case the files .EXE are used, which normally contain in the first 200H bytes, information which serve only in MS-DOS environment.
- The TX delay, which serves avoiding the PC of transmitting the data too fast to Module Z3/EV. It is used because there are no handshake lines during the parallel transmission. In the case in which there were transmission problems, try increasing this time.

## MENU UTILITY

This menu contains the commands for the direct execution of the assembling operations from the application MODZ3, Linker and binary conversion of the program.

### MENU UTILITY: Set Parameters

This command sets the parameters for the execution of the assembling operations, Linker and binary conversion.

These parameters are:

- Assembling command. It is counseled to specify a file batch (ex. MASM.BAT) where the command for starting the assembler is written. A possible file batch of this type contains the instructions:  
    `masm %1,,%1;`  
    `pause`
- Linker command. It is counseled to specify a file batch (ex. LINK.BAT) where the command for starting the link is written. A possible file batch of this type contains the instructions:  
    `link %1;`  
    `pause`
- Binary conversion command. It is counseled to specify a file batch (ex. EXE2BIN.BAT) where the command for starting the conversion programs is written. A possible file batch of this type contains the instructions:  
    `exe2bin %1.exe %1.bin`  
    `pause`

### MENU UTILITY: Assembler

Starts automatically the assembling command giving as parameter the name of the active program.

### MENU UTILITY: Linker

Starts automatically the linker command giving as parameter the name of the active program.

### **MENU UTILITY: Convert in Binary form**

Starts automatically the binary conversion command giving as parameter the name of the active program.

The operative system MS-DOS normally comprises a program (EXE2BIN.EXE) which has this finality .

*In some computers the program EXE2BIN is not available in the DOS version installed.*

In this case the binary conversion is abandoned and the file .EXE is used directly (remember in this case to skip the first 512 bytes of the file).

### **MENU UTILITY: Program List**

This command opens a window on the video, where it's possible visualizing the list of the assembled program (obviously if the file generated by the assembler NomeProg.LST is present).

### **MENU UTILITY: Program Binary Code**

This command opens a window on the video, where it's possible visualizing the binary code of the active obviously if the file NomeProg.BIN is present).

These information are important since they are the ones to be inserted in the memory of Module Z3, whenever the program is charged directly from the keyboard.

The codes are visualized starting from address 0000H.

*When they have to be inserted into Module Z3, it's necessary to start with the address where the program will be actually put (normally 0000:0800H).*

## D. DATA SHEET OF THE MICROPROCESSOR 386EX

The datasheet of the Microprocessor 386EX is shown next



### ADVANCE INFORMATION

## Intel386™ EX EMBEDDED MICROPROCESSOR

- **Static Intel386™ CPU Core**
  - Low Power Consumption
  - Operating Power Supply  
EXTB: 2.7V to 3.6V  
EXTC: 4.5V to 5.5V
  - Operating Frequency  
20 MHz EXTB at 2.7V to 3.6V  
25 MHz EXTB at 3.0V to 3.6V;  
25/33 MHz EXTC at 4.5V to 5.5V
- **Transparent Power-management System Architecture**
  - Intel System Management Mode Architecture Extension for Truly Compatible Systems
  - Power Management Transparent to Operating Systems and Application Programs
  - Programmable Power-management Modes
- **Powerdown Mode**
  - Clock Stopping at Any Time
  - Only 10–20 µA Typical CPU Sink Current
- **Full 32-bit Internal Architecture**
  - 8-, 16-, 32-bit Data Types
  - 8 General Purpose 32-bit Registers
- **Runs Intel386 Architecture Software in a Cost-effective 16-bit Hardware Environment**
  - Runs Same Applications and Operating Systems as the Intel386 SX and Intel386 DX Processors
  - Object Code Compatible with 8086, 80186, 80286, and Intel386 Processors
- **High-performance 16-bit Data Bus**
  - Two-clock Bus Cycles
  - Address Pipelining Allows Use of Slower, Inexpensive Memories
- **Extended Temperature Range**
- **Integrated Memory Management Unit**
  - Virtual Memory Support
  - Optional On-chip Paging
  - 4 Levels of Hardware-enforced Protection
  - MMU Fully Compatible with MMUs of the 80286 and Intel386 DX Processors
- **Virtual 8086 Mode Allows Execution of 8086 Software in a Protected and Paged System**
- **Large Uniform Address Space**
  - 64 Megabyte Physical
  - 64 Terabyte Virtual
  - 4 Gigabyte Maximum Segment Size
- **On-chip Debugging Support Including Breakpoint Registers**
- **Complete System Development Support**
- **High Speed CHMOS Technology**
- **Two Package Types**
  - 132-pin Plastic Quad Flatpack
  - 144-pin Thin Quad Flatpack
- **Integrated Peripheral Functions**
  - Clock and Power Management Unit
  - Chip-select Unit
  - Interrupt Control Unit
  - Timer/Counter Unit
  - Watchdog Timer Unit
  - Asynchronous Serial I/O Unit
  - Synchronous Serial I/O Unit
  - Parallel I/O Unit
  - DMA and Bus Arbiter Unit
  - Refresh Control Unit
  - JTAG-compliant Test-logic Unit

### IMPORTANT

This datasheet applies to devices marked EXTB and EXTC. If you require information about devices marked EXSA or EXTA, refer to a previous revision of this datasheet, order number 272420-004.

Intel386™ EX EMBEDDED MICROPROCESSOR



2.0 PIN ASSIGNMENT

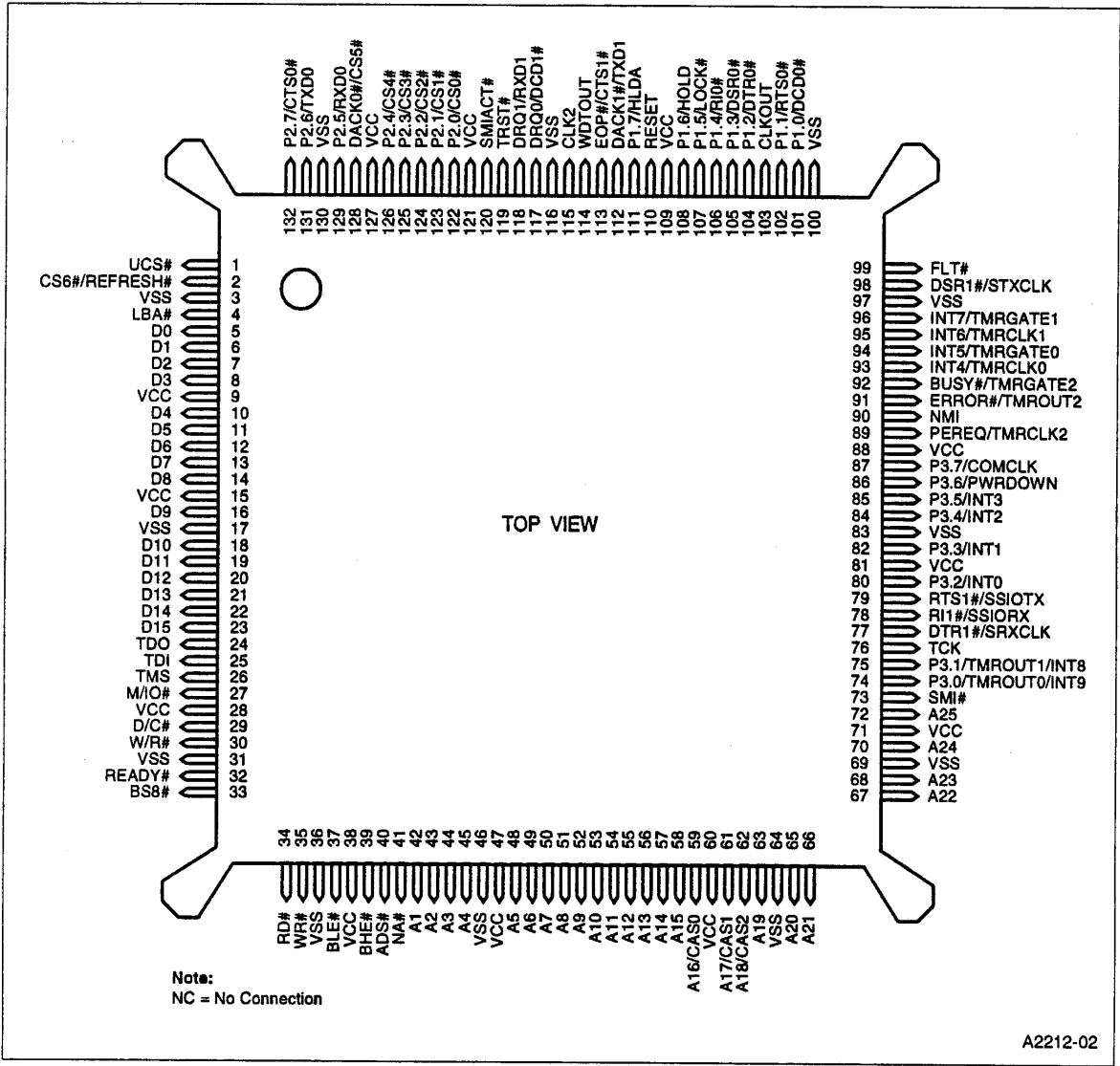


Figure 2. Intel386™ EX Embedded Processor 132-Pin PQFP Pin Assignment





## Intel386™ EX EMBEDDED MICROPROCESSOR

Table 1. 132-Pin PQFP Pin Assignment

Pin	Symbol	Pin	Symbol	Pin	Symbol	Pin	Symbol
1	UCS#	34	RD#	67	A22	100	V <sub>ss</sub>
2	CS6#/REFRESH#	35	WR#	68	A23	101	P1.0/DCD0#
3	V <sub>ss</sub>	36	V <sub>ss</sub>	69	V <sub>ss</sub>	102	P1.1/RTS0#
4	LBA#	37	BLE#	70	A24	103	CLKOUT
5	D0	38	V <sub>cc</sub>	71	V <sub>cc</sub>	104	P1.2/DTR0#
6	D1	39	BHE#	72	A25	105	P1.3/DSR0#
7	D2	40	ADS#	73	SMI#	106	P1.4/RI0#
8	D3	41	NA#	74	P3.0/TMROUT0/INT9	107	P1.5/LOCK#
9	V <sub>cc</sub>	42	A1	75	P3.1/TMROUT1/INT8	108	P1.6/HOLD
10	D4	43	A2	76	TCK	109	V <sub>cc</sub>
11	D5	44	A3	77	DTR1#/SRXCLK	110	RESET
12	D6	45	A4	78	RI1#/SSIORX	111	P1.7/HLDA
13	D7	46	V <sub>ss</sub>	79	RTS1#/SSIoTX	112	DACK1#/TXD1
14	D8	47	V <sub>cc</sub>	80	P3.2/INT0	113	EOP#/CTS1#
15	V <sub>cc</sub>	48	A5	81	V <sub>cc</sub>	114	WDTOUT
16	D9	49	A6	82	P3.3/INT1	115	CLK2
17	V <sub>ss</sub>	50	A7	83	V <sub>ss</sub>	116	V <sub>ss</sub>
18	D10	51	A8	84	P3.4/INT2	117	DRQ0/DCD1#
19	D11	52	A9	85	P3.5/INT3	118	DRQ1/RXD1
20	D12	53	A10	86	P3.6/PWRDOWN	119	TRST#
21	D13	54	A11	87	P3.7/COMCLK	120	SMIACT#
22	D14	55	A12	88	V <sub>cc</sub>	121	V <sub>cc</sub>
23	D15	56	A13	89	PEREQ/TMRCLK2	122	P2.0/CS0#
24	TDO	57	A14	90	NMI	123	P2.1/CS1#
25	TDI	58	A15	91	ERROR#/TMROUT2	124	P2.2/CS2#
26	TMS	59	A16/CAS0	92	BUSY#/TMRGATE2	125	P2.3/CS3#
27	M/IO#	60	V <sub>cc</sub>	93	INT4/TMRCLK0	126	P2.4/CS4#
28	V <sub>cc</sub>	61	A17/CAS1	94	INT5/TMRGATE0	127	V <sub>cc</sub>
29	D/C#	62	A18/CAS2	95	INT6/TMRCLK1	128	DACK0#/CS5#
30	W/R#	63	A19	96	INT7/TMRGATE1	129	P2.5/RXD0
31	V <sub>ss</sub>	64	V <sub>ss</sub>	97	V <sub>ss</sub>	130	V <sub>ss</sub>
32	READY#	65	A20	98	DSR1#/STXCLK	131	P2.6/TXD0
33	BS8#	66	A21	99	FLT#	132	P2.7/CTS0#

## ADVANCE INFORMATION

## Intel386™ EX EMBEDDED MICROPROCESSOR



## 3.0 PIN DESCRIPTION

Table 4 lists the Intel386 EX embedded processor pin descriptions. Table 3 defines the abbreviations used in the Type and Output States columns of Table 4.

Table 3. Pin Type and Output State Nomenclature

Symbol	Description
<b>Pin Type</b>	
#	The named signal is active low.
I	Standard TTL input signal.
O	Standard CMOS output signal.
I/O	Input and output signal.
I/OD	Input and open-drain output signal.
ST	Schmitt-triggered input signal.
P	Power pin.
G	Ground pin.
<b>Output State</b>	
H(1)	Output driven to $V_{CC}$ during Bus Hold
H(0)	Output driven to $V_{SS}$ during Bus Hold
H(Z)	Output floats during Bus Hold
H(Q)	Output remains active during Bus Hold
H(X)	Output retains current state during Bus Hold
R(WH)	Output Weakly Held at $V_{CC}$ during Reset
R(WL)	Output Weakly Held at $V_{SS}$ during Reset
R(1)	Output driven to $V_{CC}$ during Reset
R(0)	Output driven to $V_{SS}$ during Reset
R(Z)	Output floats during Reset
R(Q)	Output remains active during Reset
R(X)	Output retains current state during Reset
I(1) <sup>Note 1</sup>	Output driven to $V_{CC}$ during Idle Mode
I(0)	Output driven to $V_{SS}$ during Idle Mode
I(Z)	Output floats during Idle Mode
I(Q)	Output remains active during Idle Mode
I(X)	Output retains current state during Idle Mode
P(1)	Output driven to $V_{CC}$ during Powerdown Mode
P(0)	Output driven to $V_{SS}$ during Powerdown Mode
P(Z)	Output floats during Powerdown Mode
P(Q)	Output remains active during Powerdown Mode
P(X)	Output retains current state during Powerdown Mode

**NOTE:**

1. The idle mode output states assume that no internal bus master (DMA or RCU) has control of the bus during idle mode



## Intel386™ EX EMBEDDED MICROPROCESSOR

Table 4. Intel386™ EX Microprocessor Pin Descriptions (Sheet 1 of 7)

Symbol	Type	Output States	Name and Function
A25:1	O	H(Z) R(1) I(1) P(1)	<b>Address Bus</b> outputs physical memory or port I/O addresses. These signals are valid when ADS# is active and remain valid until the next T1, T2P, or T1. During HOLD cycles they are driven to a high-impedance state. A18:16 are multiplexed with CAS2:0.
ADS#	O	H(Z) R(1) I(1) P(1)	<b>Address Status</b> indicates that the processor is driving a valid bus-cycle definition and address (W/R#, D/C#, M/IO#, A25:1, BHE#, BLE#) onto its pins.
BHE#	O	H(Z) R(0) I(X) P(0)	<b>Byte High Enable</b> indicates that the processor is transferring a high data byte.
BLE#	O	H(Z) R(0) I(X) P(1)	<b>Byte Low Enable</b> indicates that the processor is transferring a low data byte.
BS8#	I		<b>Bus Size</b> indicates that an 8-bit device is currently being addressed.
BUSY#	I		<b>Busy</b> indicates that the math coprocessor is busy. If BUSY# is sampled LOW at the falling edge of RESET, the processor performs an internal self test. BUSY# is multiplexed with TMRGATE2 and has a temporary weak pull-up resistor.
CAS2:0	O	H(Z) R(1) I(1) P(1)	<b>Cascade Address</b> carries the slave address information from the 8259A master interrupt module during interrupt acknowledge bus cycles. CAS2:0 are multiplexed with A18:16.
CLK2	ST		<b>Clock Input</b> is connected to an external clock that provides the fundamental timing for the device.
CLKOUT	O	H(Q) R(Q) I(Q) P(0)	<b>CLKOUT</b> is a PH1P clock output.
COMCLK	I		<b>Serial Communications Baud Clock</b> is an alternate clock source for the asynchronous serial ports. COMCLK is multiplexed with P3.7 and has a temporary weak pull-down resistor.

## NOTES:

1. X if clock source is internal; Q if clock source is external
2. Q if JTAG unit is shifting out data, Z if it is not

## Intel386™ EX EMBEDDED MICROPROCESSOR



Table 4. Intel386™ EX Microprocessor Pin Descriptions (Sheet 2 of 7)

Symbol	Type	Output States	Name and Function
CS4:0#  CS6:5#	O	H(1) R(WH) I(Q) P(X) H(1) R(1) I(Q) P(X)	<b>Chip-selects</b> are activated when the address of a memory or I/O bus cycle is within the address region programmed by the user. They are multiplexed as follows: CS6# with REFRESH#, CS5# with DACK0#, and CS4:0# with P2.4:0.
CTS1:0#	I		<b>Clear to Send SIO1 and SIO0</b> prevent the transmission of data to the asynchronous serial port's RXD1 and RXD0 pins, respectively. CTS1# is multiplexed with EOP#, and CTS0# is multiplexed with P2.7. CTS1# requires an external pull-up resistor. Both have temporary weak pull-up resistors.
D15:0	I/O	H(Z) R(Z) P(Z)	<b>Data Bus</b> inputs data during memory read, I/O read, and interrupt acknowledge cycles and outputs data during memory and I/O write cycles. During writes, this bus is driven during phase 2 of T1 and remains active until phase 2 of the next T1, T1P, or Ti. During reads, data is latched on the falling edge of phase 2.
DACK1:0#	O	H(1) R(1) I(Q) P(X)	<b>DMA Acknowledge 1 and 0</b> signal to an external device that the processor has acknowledged the corresponding DMA request and is relinquishing the bus. DACK1# is multiplexed with TXD1, and DACK0# is multiplexed with CS5#.
D/C#	O	H(Z) R(1) I(0) P(0)	<b>Data/Control</b> indicates whether the current bus cycle is a data cycle (memory or I/O read or write) or a control cycle (interrupt acknowledge, halt, or code fetch).
DCD1:0	I		<b>Data Carrier Detect SIO1 and SIO0</b> indicate that the modem or data set has detected the corresponding asynchronous serial channel's data carrier. DCD1# is multiplexed with DRQ0, and DCD0# is multiplexed with P1.0 and has a temporary weak pull-up resistor.
DRQ1:0	I		<b>DMA External Request 1 and 0</b> indicate that a peripheral requires DMA service. DRQ1 is multiplexed with RXD1, and DRQ0 is multiplexed with DCD1#.
DSR1:0#	I		<b>Data Set Ready SIO1 and SIO0</b> indicate that the modem or data set is ready to establish a communication link with the corresponding asynchronous serial channel. DSR1# is multiplexed with STXCLK and has a permanent weak pull-up resistor, and DSR0# is multiplexed with P1.3 and has a temporary weak pull-up resistor.

**NOTES:**

1. X if clock source is internal; Q if clock source is external
2. Q if JTAG unit is shifting out data, Z if it is not



## Intel386™ EX EMBEDDED MICROPROCESSOR

Table 4. Intel386™ EX Microprocessor Pin Descriptions (Sheet 3 of 7)

Symbol	Type	Output States	Name and Function
DTR1:0#	O	H(X) R(WH) I(X) P(X)	<b>Data Terminal Ready SIO1 and SIO0</b> indicate that the corresponding asynchronous serial channel is ready to establish a communication link with the modem or data set. DTR1# is multiplexed with SRXCLK, and DTR0# is multiplexed with P1.2.
EOP#	I/OD	H(Z) R(WH) I(Z) P(Z)	<b>End of Process</b> indicates that the processor has reached terminal count during a DMA transfer. An external device can also pull this pin LOW. EOP# is multiplexed with CTS1#.
ERROR#	I		<b>Error</b> indicates that the math coprocessor has an error condition. ERROR# is multiplexed with TMROUT2 and has a temporary weak pull-up resistor.
FLT#	I		<b>Float</b> forces all bidirectional and output signals except TDO to a high-impedance state. It has a permanent weak pull-up resistor. This pin should be a no-connect if not used.
HLDA	O	H(1) R(WL) I(Q) P(X)	<b>Bus Hold Acknowledge</b> indicates that the processor has surrendered control of its local bus to another bus master. HLDA is multiplexed with P1.7.
HOLD	I		<b>Bus Hold Request</b> allows another bus master to request control of the local bus. HLDA active indicates that bus control has been granted. HOLD is multiplexed with P1.6. It has a temporary weak pull-down resistor.
INT9:0	I		<b>Interrupt Requests</b> are maskable inputs that cause the CPU to suspend execution of the current program and then execute an interrupt acknowledge cycle. They are multiplexed as follows: INT9 with TMROUT0 and P3.0, INT8 with TMROUT1 and P3.1, INT7 with TMRGATE1, INT6 with TMRCLK1, INT5 with TMRGATE0, INT4 with TMRCLK0, and INT3:0 with P3.5:2. INT9, INT8, and INT3:0 have temporary weak pull-down resistors.
LBA#	O	H(1) R(1) I(Q) P(X)	<b>Local Bus Access</b> is asserted whenever the processor provides the READY# signal to terminate a bus transaction. This occurs when an internal peripheral address is accessed or when the chip-select unit provides the READY# signal.
LOCK#	O	H(Z) R(WH) I(X) P(X)	<b>Bus Lock</b> prevents other bus masters from gaining control of the system bus. LOCK# is multiplexed with P1.5.
M/IO#	O	H(Z) R(0) I(1) P(1)	<b>Memory/IO</b> Indicates whether the current bus cycle is a memory cycle or an I/O cycle. When M/IO# is HIGH, the bus cycle is a memory cycle; when M/IO# is LOW, the bus cycle is an I/O cycle.

## NOTES:

1. X if clock source is internal; Q if clock source is external
2. Q if JTAG unit is shifting out data, Z if it is not

## ADVANCE INFORMATION

## Intel386™ EX EMBEDDED MICROPROCESSOR



Table 4. Intel386™ EX Microprocessor Pin Descriptions (Sheet 4 of 7)

Symbol	Type	Output States	Name and Function
NA#	I		<b>Next Address</b> requests address pipelining.
NMI	ST		<b>Nonmaskable Interrupt Request</b> is a non-maskable input that causes the CPU to suspend execution of the current program and execute an interrupt acknowledge cycle.
PEREQ	I		<b>Processor Extension Request</b> indicates that the math coprocessor has data to transfer to the processor. PEREQ is multiplexed with TMRCLK2 and has a temporary weak pull-down resistor.
P1.5:0  P1.7:6	I/O	H(X) R(WH) I(X) P(X) H(X) R(WL) I(X) P(X)	<b>Port 1, Pins 7:0</b> are multipurpose bidirectional port pins. They are multiplexed as follows: P1.7 with HLDA, P1.6 with HOLD, P1.5 with LOCK#, P1.4 with RI0#, P1.3 with DSR0#, P1.2 with DTR0#, P1.1 with RTS0#, and P1.0 with DCD0#.
P2.7,4:0  P2.6:5	I/O	H(X) R(WH) I(X) P(X) H(X) R(WL) I(X) P(X)	<b>Port 2, Pins 7:0</b> are multipurpose bidirectional port pins. They are multiplexed as follows: P2.7 with CTS0#, P2.6 with TXD0, P2.5 with RXD0, and P2.4:0 with CS4:0#.
P3.7:0	I/O	H(X) R(WL) I(X) P(X)	<b>Port 3, Pins 7:0</b> are multipurpose bidirectional port pins. They are multiplexed as follows: P3.7 with COMCLK, P3.6 with PWRDOWN, P3.5:2 with INT3:0, and P3.1:0 with TMROUT1:0 and INT8:9.
PWRDOWN	O	H(Q) R(WL) I(X) P(1)	<b>Powerdown</b> indicates that the processor is in powerdown mode. PWRDOWN is multiplexed with P3.6.
RD#	O	H(1) R(1) I(1) P(1)	<b>Read Enable</b> indicates that the current bus cycle is a read cycle.
READY#	I/O	H(Z) R(Z) I(Z) P(Z)	<b>Ready</b> indicates that the current bus transaction has completed. An external device or an internal signal can drive READY#. Internally, the chip-select wait-state logic can generate the ready signal and drive the READY# pin active.
RESET	ST		<b>Reset</b> suspends any operation in progress and places the processor into a known reset state.

## NOTES:

1. X if clock source is internal; Q if clock source is external
2. Q if JTAG unit is shifting out data, Z if it is not



## Intel386™ EX EMBEDDED MICROPROCESSOR

Table 4. Intel386™ EX Microprocessor Pin Descriptions (Sheet 5 of 7)

Symbol	Type	Output States	Name and Function
REFRESH#	O	H(1) R(1) I(Q) P(X)	<b>Refresh</b> indicates that the current bus cycle is a refresh cycle. REFRESH# is multiplexed with CS6#.
RI1:0#	I		<b>Ring Indicator SIO1 and SIO0</b> indicate that the modem or data set has received a telephone ringing signal. RI1# is multiplexed with SSIORX, and RI0# is multiplexed with P1.4 and has a temporary weak pull-up resistor.
RTS1  RTS0	O	H(X) R(WL) I(X) P(X) H(X) R(WH) I(X) P(X)	<b>Request-to-send SIO1 and SIO0</b> indicate that corresponding asynchronous serial channel is ready to exchange data with the modem or data set. RTS1# is multiplexed with SSIO TX, and RTS0# is multiplexed with P1.1.
RXD1:0	I		<b>Receive Data SIO1 and SIO0</b> accept serial data from the modem or data set to the corresponding asynchronous serial channel. RXD1 is multiplexed with DRQ1, and RXD0 is multiplexed with P2.5 and has a temporary weak pull-down resistor.
SMI#	ST		<b>System Management Interrupt</b> invokes System Management Mode (SMM). SMI# is the highest priority external interrupt. It is latched on its falling edge and forces the CPU into SMM upon completion of the current instruction. SMI# is recognized on an instruction boundary and at each iteration for repeat string instructions. SMI# cannot interrupt LOCKed bus cycles or a currently executing SMM. When the processor receives a second SMI# while in SMM, it latches the second SMI# on the SMI# falling edge. However, the processor must exit SMM by executing a resume instruction (RSM) before it can service the second SMI#. SMI# has a permanent weak pull-up resistor.
SMIACK#	O	H(1) R(1) I(X) P(X)	<b>System Management Interrupt Active</b> indicates that the processor is operating in System Management Mode (SMM). It is asserted when the processor initiates an SMM sequence and remains asserted (LOW) until the processor executes the resume instruction (RSM).
SRXCLK	I/O	H(Q) R(WH) I(Q) P(X)/P(Q) <sup>Note 1</sup>	<b>SSIO Receive Clock</b> synchronizes data being accepted by the synchronous serial port. SRXCLK is multiplexed with DTR1#.
SSIORX	I		<b>SSIO Receive Serial Data</b> accepts serial data (most-significant bit first) being sent to the synchronous serial port. SSIORX is multiplexed with RI1#.

## NOTES:

1. X if clock source is internal; Q if clock source is external
2. Q if JTAG unit is shifting out data, Z if it is not

## ADVANCE INFORMATION

11

## Intel386™ EX EMBEDDED MICROPROCESSOR



Table 4. Intel386™ EX Microprocessor Pin Descriptions (Sheet 6 of 7)

Symbol	Type	Output States	Name and Function
SSIOTX	O	H(Q) R(WL) I(Q) P(X)/P(Q) <sup>Note 1</sup>	<b>SSIO Transmit Serial Data</b> sends serial data (most-significant bit first) from the synchronous serial port. SSIOTX is multiplexed with RTS1#.
STXCLK	I/O	H(Q) R(WH) I(Q) P(X)/P(Q) <sup>Note 1</sup>	<b>SSIO Transmit Clock</b> synchronizes data being sent by the synchronous serial port. STXCLK is multiplexed with DSR1.
TCK	I		<b>TAP (Test Access Port) Controller Clock</b> provides the clock input for the JTAG logic. It has a permanent weak pull-up resistor.
TDI	I		<b>TAP (Test Access Port) Controller Data Input</b> is the serial input for test instructions and data. It has a permanent weak pull-up resistor.
TDO	O	H(Z)/H(Q) <sup>Note 2</sup> R(Z)/R(Q) <sup>Note 2</sup> I(Z)/I(Q) <sup>Note 2</sup> P(Z)/P(Q) <sup>Note 2</sup>	<b>TAP (Test Access Port) Controller Data Output</b> is the serial output for test instructions and data.
TMRCLK2:0	I		<b>Timer/Counter Clock Inputs</b> can serve as external clock inputs for the corresponding timer/counters. (The timer/counters can also be clocked internally.) They are multiplexed as follows: TMRCLK2 with PEREQ, TMRCLK1 with INT6, and TMRCLK0 with INT4. TMRCLK2 has a temporary weak pull-down resistor.
TMRGATE2:0	I		<b>Timer/Counter Gate Inputs</b> can control the corresponding timer/counter's counting (enable, disable, or trigger, depending on the programmed mode). They are multiplexed as follows: TMRGATE2 with BUSY#, TMRGATE1 with INT7, and TMRGATE0 with INT5. TMRGATE2 has a temporary weak pull-up resistor.
TMROUT2 TMROUT1:0	O	H(Q) R(WH) I(Q) P(X)/P(Q) <sup>Note 1</sup> H(Q) R(WL) I(Q) P(X)/P(Q) <sup>Note 1</sup>	<b>Timer/Counter Outputs</b> provide the output of the corresponding timer/counter. The form of the output depends on the programmed mode. They are multiplexed as follows: TMROUT2 with ERROR#, TMROUT1 with P3.1 and INT8, and TMROUT0 with P3.0 and INT9.
TMS	I		<b>TAP (Test Access Port) Controller Mode Select</b> controls the sequence of the TAP controller's states. It has a permanent weak pull-up resistor.
TRST#	ST		<b>TAP (Test Access Port) Controller Reset</b> resets the TAP controller at power-up and each time it is activated. It has a permanent weak pull-up resistor.

## NOTES:

1. X if clock source is internal; Q if clock source is external
2. Q if JTAG unit is shifting out data, Z if it is not





## Intel386™ EX EMBEDDED MICROPROCESSOR

Table 4. Intel386™ EX Microprocessor Pin Descriptions (Sheet 7 of 7)

Symbol	Type	Output States	Name and Function
TXD1  TXD0	O	H(Q) R(1) I(Q) P(X)/P(Q) <sup>Note 1</sup> H(Q) R(WL) I(Q) P(X)/P(Q) <sup>Note 1</sup>	<b>Transmit Data SIO1 and SIO0</b> transmit serial data from the individual serial channels. TXD1 is multiplexed with DACK1#, and TXD0 is multiplexed with P2.6.
UCS#	O	H(1) R(0) I(Q) P(X)	<b>Upper Chip-select</b> is activated when the address of a memory or I/O bus cycle is within the address region programmed by the user.
V <sub>CC</sub>	P		<b>System Power</b> provides the nominal DC supply input. This pin is connected externally to a V <sub>CC</sub> board plane.
V <sub>SS</sub>	G		<b>System Ground</b> provides the 0 V connection from which all inputs and outputs are measured. This pin is connected externally to a ground board plane.
WDTOUT	O	H(Q) R(0) I(Q) P(X)	<b>Watchdog Timer Output</b> indicates that the watchdog timer has expired.
W/R#	O	H(Z) R(0) I(1) P(1)	<b>Write/Read</b> indicates whether the current bus cycle is a write cycle or a read cycle. When W/R# is HIGH, the bus cycle is a write cycle; when W/R# is LOW, the bus cycle is a read cycle.
WR#	O	H(1) R(1) I(1) P(1)	<b>Write Enable</b> indicates that the current bus cycle is a write cycle.

**NOTES:**

1. X if clock source is internal; Q if clock source is external
2. Q if JTAG unit is shifting out data, Z if it is not

## Intel386™ EX EMBEDDED MICROPROCESSOR



## 4.0 FUNCTIONAL DESCRIPTION

The Intel386 EX microprocessor is a fully static, 32-bit processor optimized for embedded applications. It features low power and low voltage capabilities, integration of many commonly used DOS-type peripherals, and a 32-bit programming architecture compatible with the large software base of Intel386 processors. The following sections provide an overview of the integrated peripherals.

### 4.1 Clock Generation and Power Management Unit

The clock generation circuit includes a divide-by-two counter, a programmable divider for generating a prescaled clock (PCLK), a divide-by-two counter for generating baud-rate clock inputs, and Reset circuitry. The CLK2 input provides the fundamental timing for the chip. It is divided by two internally to generate a 50% duty cycle Phase1 (PH1) and Phase 2 (PH2) for the core and integrated peripherals. For power management, separate clocks are routed to the core (PH1C/PH2C) and the peripheral modules (PH1P/PH2P). To help synchronize with external devices, the PH1P clock is provided on the CLKOUT output pin.

Two Power Management modes are provided for flexible power-saving options. During Idle mode, the clocks to the CPU core are frozen in a known state (PH1C low and PH2C high), while the clocks to the peripherals continue to toggle. In Powerdown mode, the clocks to both core and peripherals are frozen in a known state (PH1C low and PH2C high). The Bus Interface Unit will not honor any DMA, DRAM refresh, or HOLD requests in Powerdown mode because the clocks to the entire device are frozen.

### 4.2 Chip-select Unit

The Chip-select Unit (CSU) decodes bus cycle address and status information and enables the appropriate chip-selects. The individual chip-selects become valid in the same bus state as the address and become inactive when either a new address is selected or the current bus cycle is complete.

The CSU is divided into eight separate chip-select regions, each of which can enable one of the eight chip-select pins. Each chip-select region can be mapped into memory or I/O space. A memory-mapped chip-select region can start on any  $2^{(n+1)}$  Kbyte address location (where  $n = 0-15$ , depending upon the mask register). An I/O-mapped chip-select region can start on any  $2^{(n+1)}$  byte address location (where  $n = 0-15$ , depending upon the mask register). The size of the region is also dependent upon the mask used.

### 4.3 Interrupt Control Unit

The Intel386 EX processor's Interrupt Control Unit (ICU) contains two 8259A modules connected in a cascade mode. These modules are similar to the industry-standard 8259A architecture.

The Interrupt Control Unit directly supports up to ten external (INT9:0) and up to eight internal interrupt request signals. Pending interrupt requests are posted in the Interrupt Request Registers, which contain one bit for each interrupt request signal. When an interrupt request is asserted, the corresponding Interrupt Request Register bit is set. The 8259A modules can be programmed to recognize either an active-high level or a positive transition on the interrupt request lines. An internal Priority Resolver decides which pending interrupt request (if more than one exists) is the highest priority, based on the programmed operating mode. The Priority Resolver controls the single interrupt request line to the CPU. The Priority Resolver's default priority scheme places the master interrupt controller's IR0 as the highest priority and the master's IR7 as the lowest. The priority can be modified through software.

Besides the ten interrupt request inputs available to the Intel386 EX microprocessor, additional interrupts can be supported by cascaded external 8259A modules. Up to four external 8259A units can be cascaded to the master through connections to the INT3:0 pins. In this configuration, the interrupt acknowledge (INTA#) signal can be decoded externally using the ADS#, D/C#, W/R#, and M/IO# signals.



## Intel386™ EX EMBEDDED MICROPROCESSOR

### 4.4 Timer/Counter Unit

The Timer/Counter Unit (TCU) on the Intel386 EX microprocessor has the same basic functionality as the industry-standard 82C54 counter/timer. The TCU provides three independent 16-bit counters, each capable of handling clock inputs up to 8 MHz. This maximum frequency must be considered when programming the input clocks for the counters. Six programmable timer modes allow the counters to be used as event counters, elapsed-time indicators, programmable one-shots, and in many other applications. All modes are software programmable.

### 4.5 Watchdog Timer Unit

The Watchdog Timer (WDT) unit consists of a 32-bit down-counter that decrements every PH1P cycle, allowing up to 4.3 billion count intervals. The WDTOUT pin is driven high for sixteen CLK2 cycles when the down-counter reaches zero (the WDT times out). The WDTOUT signal can be used to reset the chip, to request an interrupt, or to indicate to the user that a ready-hang situation has occurred. The down-counter can also be updated with a user-defined 32-bit reload value under certain conditions. Alternatively, the WDT unit can be used as a bus monitor or as a general-purpose timer.

### 4.6 Asynchronous Serial I/O Unit

The Intel386 EX microprocessor's asynchronous Serial I/O (SIO) unit is a Universal Asynchronous Receiver/ Transmitter (UART). Functionally, it is equivalent to the National Semiconductor NS16450 and INS8250. The Intel386 EX embedded processor contains two full-duplex, asynchronous serial channels.

The SIO unit converts serial data characters received from a peripheral device or modem to parallel data and converts parallel data characters received from the CPU to serial data. The CPU can read the status of the serial port at any time during its operation. The status information includes the type and condition of the transfer operations being performed and any errors (parity, framing, overrun, or break interrupt).

Each asynchronous serial channel includes full modem control support (CTS#, RTS#, DSR#, DTR#, RI#, and DCD#) and is completely programmable. The programmable options include character length (5, 6, 7, or 8 bits), stop bits (1, 1.5, or 2), and parity (even, odd, forced, or none). In addition, it contains a programmable baud-rate generator capable of clock rates from 0 to 512 Kbaud.

### 4.7 Synchronous Serial I/O Unit

The Synchronous Serial I/O (SSIO) unit provides for simultaneous, bidirectional communications. It consists of a transmit channel, a receive channel, and a dedicated baud-rate generator. The transmit and receive channels can be operated independently (with different clocks) to provide non-lockstep, full-duplex communications; either channel can originate the clocking signal (Master Mode) or receive an externally generated clocking signal (Slave Mode).

The SSIO provides numerous features for ease and flexibility of operation. With a maximum clock input of CLK2/4 to the baud-rate generator, the SSIO can deliver a baud rate of up to 8.25 Mbits per second with a processor clock of 33 MHz. Each channel is double buffered. The two channels share the baud-rate generator and a multiply-by-two transmit and receive clock. The SSIO supports 16-bit serial communications with independently enabled transmit and receive functions and gated interrupt outputs to the interrupt controller.

### 4.8 Parallel I/O Unit

The Intel386 EX microprocessor has three 8-bit, general-purpose I/O ports. All port pins are bidirectional, with TTL-level inputs and CMOS-level outputs. All pins have both a standard operating mode and a peripheral mode (a multiplexed function), and all have similar sets of control registers located in I/O address space.

### 4.9 DMA and Bus Arbiter Unit

The Intel386 EX microprocessor's DMA controller is a two-channel DMA; each channel operates independently of the other. Within the operation of the individual channels, several different data

## ADVANCE INFORMATION

**Intel386™ EX EMBEDDED MICROPROCESSOR**

transfer modes are available. These modes can be combined in various configurations to provide a very versatile DMA controller. Its feature set has enhancements beyond the 8237 DMA family; however, it can be configured such that it can be used in an 8237-like mode. Each channel can transfer data between any combination of memory and I/O with any combination (8 or 16 bits) of data path widths. An internal temporary register that can disassemble or assemble data to or from either an aligned or a nonaligned destination or source optimizes bus bandwidth.

The bus arbiter, a part of the DMA controller, works much like the priority resolving circuitry of a DMA. It receives service requests from the two DMA channels, the external bus master, and the DRAM Refresh Control Unit. The bus arbiter requests bus ownership from the core and resolves priority issues among all active requests when bus mastership is granted.

Each DMA channel consists of three major components: the Requestor, the Target, and the Byte Count. These components are identified by the contents of programmable registers that define the memory or I/O device being serviced by the DMA. The Requestor is the device that requires and requests service from the DMA controller. Only the Requestor is considered capable of initializing or terminating a DMA process. The Target is the device with which the Requestor wishes to communicate. The DMA process considers the Target a slave that is incapable of controlling the process. The Byte Count dictates the amount of data that must be transferred.

#### 4.10 Refresh Control Unit

The Refresh Control Unit (RCU) simplifies dynamic memory controller design with its integrated address and clock counters. Integrating the RCU into the processor allows an external DRAM controller to use chip-selects, wait state logic, and status lines.

The Refresh Control Unit:

- Provides a programmable-interval timer
- Provides the bus arbitration logic to gain control of the bus to run refresh cycles
- Contains the logic to generate row addresses to refresh DRAM rows individually
- Contains the logic to signal the start of a refresh cycle

The RCU contains a 13-bit address counter that forms the refresh address, supporting DRAMs with up to 13 rows of memory cells (13 refresh address bits). This includes all practical DRAM sizes for the Intel386 EX microprocessor's 64 Mbyte address space.

#### 4.11 JTAG Test-logic Unit

The JTAG Test-logic Unit provides access to the device pins and to a number of other testable areas on the device. It is fully compliant with the IEEE 1149.1 standard and thus interfaces with five dedicated pins: TRST#, TCK, TMS, TDI, and TDO. It contains the Test Access Port (TAP) finite-state machine, a 4-bit instruction register, a 32-bit identification register, and a single-bit bypass register. The test-logic unit also contains the necessary logic to generate clock and control signals for the Boundary Scan chain.

Since the test-logic unit has its own clock and reset signals, it can operate autonomously. While the rest of the microprocessor is in Reset or Powerdown, the JTAG unit can read or write various register chains.

**E. ADDITION INSTRUCTIONS 386EX**

The tables referring to the addition of the instructions of the microprocessor 386EX.



## APPENDIX E

## INSTRUCTION SET SUMMARY

This appendix provides reference information for the Intel386™ processor family instruction set.

The appendix is organized as follows:

- Instruction Encoding and Clock Count Summary (see below)
- Instruction Encoding (page E-22)

### E.1 INSTRUCTION ENCODING AND CLOCK COUNT SUMMARY

To calculate elapsed time for an instruction, multiply the instruction clock count, as listed in Table E-1, by the processor clock period (e.g., 62.5 ns for 16 MHz).

Instruction clock count assumptions:

- The instruction has been prefetched, decoded, and is ready for execution.
- Bus cycles do not require wait states.
- There are no local bus HOLD requests delaying processor access to the bus.
- No exceptions are detected during instruction execution.
- When an effective address is calculated, it does not use two general register components. One register, scaling and displacement can be used within the clock counts shown. However, when the effective address calculation uses two general register components, add 1 clock to the clock count shown.

Instruction clock count notation:

- When two clock counts are given, the smaller refers to a register operand and the larger refers to a memory operand.
- n = number of times repeated.
- m = number of components in the next instruction executed, where the entire displacement (if any) counts as one component, the entire immediate data (if any) counts as one component, and all other bytes of the instruction and prefix(es) of each count as one component.

Misaligned or 32-bit operand accesses:

- When instructions access a misaligned 16-bit operand or 32-bit operand on even address:
  - add 2\* clocks for read or write
  - add 4\*\* clocks for read and write
- When instructions access a 32-bit operand on odd address:
  - add 4\* clocks for read or write
  - add 8\*\* clocks for read and write

## Intel386™ EX EMBEDDED MICROPROCESSOR USER'S MANUAL



Wait states:

Wait states add 1 clock per wait state to instruction execution for each data access.

Table E-1 lists the instructions with their formats and execution times. The description of the notes for Table E-1 begins on page E-20. See "Instruction Encoding" on page E-22 for the definition of the terms used in this table.

**Table E-1. Instruction Set Summary (Sheet 1 of 19)**

Instruction	Format	Clock Count		Notes				
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode			
<b>GENERAL DATA TRANSFER</b>								
<b>MOV = Move</b>								
register to register/memory	<table><tr><td>1 0 0 0 1 0 0 w</td><td>mod reg r/m</td></tr></table>	1 0 0 0 1 0 0 w	mod reg r/m	2/2	2/2*	b	h	
1 0 0 0 1 0 0 w	mod reg r/m							
register/memory to register	<table><tr><td>1 0 0 0 1 0 1 w</td><td>mod reg r/m</td></tr></table>	1 0 0 0 1 0 1 w	mod reg r/m	2/4	2/4*	b	h	
1 0 0 0 1 0 1 w	mod reg r/m							
immediate to register/memory	<table><tr><td>1 1 0 0 0 1 1 w</td><td>mod 0 0 0 r/m</td></tr></table> immediate data	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	2/2	2/2*	b	h	
1 1 0 0 0 1 1 w	mod 0 0 0 r/m							
immediate to register (short form)	<table><tr><td>1 0 1 1 w reg</td><td>immediate data</td></tr></table>	1 0 1 1 w reg	immediate data	2	2			
1 0 1 1 w reg	immediate data							
memory to accumulator (short form)	<table><tr><td>1 0 1 0 0 0 0 w</td><td>full displacement</td></tr></table>	1 0 1 0 0 0 0 w	full displacement	4*	4*	b	h	
1 0 1 0 0 0 0 w	full displacement							
accumulator to memory (short form)	<table><tr><td>1 0 1 0 0 0 1 w</td><td>full displacement</td></tr></table>	1 0 1 0 0 0 1 w	full displacement	2*	2*	b	h	
1 0 1 0 0 0 1 w	full displacement							
register memory to segment register	<table><tr><td>1 0 0 0 1 1 1 0</td><td>mod sreg3 r/m</td></tr></table>	1 0 0 0 1 1 1 0	mod sreg3 r/m	2/5	22/23	b	h, i, j	
1 0 0 0 1 1 1 0	mod sreg3 r/m							
segment register to register/memory	<table><tr><td>1 0 0 0 1 1 0 0</td><td>mod sreg3 r/m</td></tr></table>	1 0 0 0 1 1 0 0	mod sreg3 r/m	2/2	2/2	b	h	
1 0 0 0 1 1 0 0	mod sreg3 r/m							
<b>MOVSBX = Move with sign extension</b>								
register from register/memory	<table><tr><td>0 0 0 0 1 1 1 1</td><td>1 0 1 1 1 1 1 w</td><td>mod reg r/m</td></tr></table>	0 0 0 0 1 1 1 1	1 0 1 1 1 1 1 w	mod reg r/m	3/6*	3/6*	b	h
0 0 0 0 1 1 1 1	1 0 1 1 1 1 1 w	mod reg r/m						
<b>MOVZXB = Move with zero extension</b>								
register from register/memory	<table><tr><td>0 0 0 0 1 1 1 1</td><td>1 0 1 1 0 1 1 w</td><td>mod reg r/m</td></tr></table>	0 0 0 0 1 1 1 1	1 0 1 1 0 1 1 w	mod reg r/m	3/6*	3/6*	b	h
0 0 0 0 1 1 1 1	1 0 1 1 0 1 1 w	mod reg r/m						
<b>PUSH = Push</b>								
register/memory	<table><tr><td>1 1 1 1 1 1 1 1</td><td>mod 1 1 0 r/m</td></tr></table>	1 1 1 1 1 1 1 1	mod 1 1 0 r/m	5/7*	7/9*	b	h	
1 1 1 1 1 1 1 1	mod 1 1 0 r/m							
register (short form)	<table><tr><td>0 1 0 1 0 reg</td><td></td></tr></table>	0 1 0 1 0 reg		2	4	b	h	
0 1 0 1 0 reg								
segment register (ES, CS, SS, or DS) (short form)	<table><tr><td>0 0 0 sreg2 1 1 0</td><td></td></tr></table>	0 0 0 sreg2 1 1 0		2	4	b	h	
0 0 0 sreg2 1 1 0								
segment register (ES, CS, SS, or DS, FS or GS)	<table><tr><td>0 0 0 0 1 1 1 1</td><td>10 sreg3 0 0 0</td></tr></table>	0 0 0 0 1 1 1 1	10 sreg3 0 0 0	2	4	b	h	
0 0 0 0 1 1 1 1	10 sreg3 0 0 0							



## INSTRUCTION SET SUMMARY

Table E-1. Instruction Set Summary (Sheet 2 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
immediate	011010s0 immediate data	2	4	b	h
<b>PUSHA</b> = Push All	01100000	18	34	b	h
<b>POP</b> = Pop					
register/memory	10001111 mod 000 r/m	5/7	7/9	b	h
register (short form)	01011 reg	6	6	b	h
segment register (ES, CS, SS, or DS) (short form)	000 sreg2 111	7	25	b	h, i, j
segment register (ES, CS, SS, or DS) FS or GS	00001111 10 sreg3 001	7	25	b	h, i, j
<b>POPA</b> = Pop all	01100001	29	35	b	h
<b>XCHG</b> = Exchange					
register/memory with register	1000011w mod reg r/m	3/5**	3/5**	b, f	f, h
register with accumulator (short form)	10010 reg	3	3		
<b>IN</b> = Input from					
fixed port	1110010w port number	14*	8*/29*		s/t, m
variable port	1110110w	15*	9*/29*		s/t, m
<b>OUT</b> = Output to					
fixed port	1110011w port number	14*	8*/29*		s/t, m
variable port	1110111w	15*	9*/29*		s/t, m
<b>LEA</b> = Load EA to register	10001101 mod reg r/m	2	2		
<b>SEGMENT CONTROL</b>					
<b>LDS</b> = Load pointer to DS	11000101 mod reg r/m	7*	26*/28*	b	h, i, j
<b>LES</b> = Load pointer to ES	11000100 mod reg r/m	7*	26*/28*	b	h, i, j
<b>LFS</b> = Load pointer to FS	00001111 10110100 mod reg r/m	7*	29*/31*	b	h, i, j
<b>LGS</b> = Load pointer to GS	00001111 10110101 mod reg r/m	7*	26*/28*	b	h, i, j
<b>LSS</b> = Load pointer to SS	00001111 10110010 mod reg r/m	7*	26*/28*	b	h, i, j
<b>FLAG CONTROL</b>					
<b>CLC</b> = Clear carry flag	11111000	2	2		



Table E-1. Instruction Set Summary (Sheet 3 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
CLD = Clear direction flag	11111100	2	2		
CLI = Clear interrupt enable flag	11111010	8	8		m
CLTS = Clear task switched flag	00001111 00000110	5	5	c	l
CMC = Complement carry flag	11110101	2	2		
LAHF = Load AH into flag	10011111	2	2		
POPF = Pop flags	10011101	5	5	b	h, n
PUSHF = Push flags	10011100	4	4	b	h
SAHF = Store AH into flags	10011110	3	3		
STC = Set carry flag	11111001	2	2		
STD = Set direction flag	11111101				
STI = Set interrupt enable flag	11111011	8	8		m
<b>ARITHMETIC INSTRUCTIONS</b>					
<b>ADD = Add</b>					
register to register	000000dw mod reg r/m	2	2		
register to memory	0000000w mod reg r/m	7**	7**	b	h
memory to register	0000001w mod reg r/m	6*	6*	b	h
immediate to register/memory	100000sw mod 000 r/m immediate data	2/7**	2/7**	b	h
immediate to accumulator (short form)	0000010w immediate data	2	2		
<b>ADC = Add with carry</b>					
register to register	000100dw mod reg r/m	2	2		
register to memory	0001000w mod reg r/m	7**	7**	b	h
memory to register	0001001w mod reg r/m	6*	6*	b	h
immediate to register/memory	100000sw mod 010 r/m immediate data	2/7**	2/7**	b	h
immediate to accumulator (short form)	0001010w immediate data	2	2		
<b>INC = Increment</b>					
register/memory	1111111w mod 000 r/m	2/6**	2/6**	b	h
register (short form)	01000 reg	2	2		





## INSTRUCTION SET SUMMARY

Table E-1. Instruction Set Summary (Sheet 4 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
<b>SUB = Subtract</b>					
register from register	001010dw mod reg r/m	2	2		
register from memory	0010100w mod reg r/m	7**	7**	b	h
memory from register	0010101w mod reg r/m	6*	6*	b	h
immediate from register/memory	100000sw mod 101 r/m	2/7**	2/7**	b	h
immediate from accumulator (short form)	0010110w immediate data	2	2		
<b>SBB = Subtract with borrow</b>					
register from register	000110dw mod reg r/m	2	2		
register from memory	0001100w mod reg r/m	7**	7**	b	h
memory from register	0001101w mod reg r/m	6*	6*	b	h
immediate from register/memory	100000sw mod 011 r/m	2/7**	2/7**	b	h
immediate from accumulator (short form)	0001110w immediate data	2	2		
<b>DEC = Decrement</b>					
register/memory	1111111w reg 001 r/m	2/6	2/6	b	h
register (short form)	01001 reg	2	2		
<b>CMP = Compare</b>					
register with register	001110dw mod reg r/m	2	2		
memory with register	0011100w mod reg r/m	5*	5*	b	h
register with memory	0011101w mod reg r/m	6*	6*	b	h
immediate with register/memory	100000sw mod 111 r/m	2/5*	2/5*	b	h
immediate with accumulator (short form)	0011110w immediate data	2	2		
<b>NEG = Change sign</b>	1111011w mod 011 r/m	2/6*	2/6*	b	h
<b>AAA = ASCII adjust for addition</b>	00110111	4	4		
<b>AAS = ASCII adjust for subtraction</b>	00111111	4	4		
<b>DAA = Decimal adjust for addition</b>	00100111	4	4		
<b>DAS = Decimal adjust for subtraction</b>	00101111	4	4		



Table E-1. Instruction Set Summary (Sheet 5 of 19)

Instruction	Format	Clock Count		Notes				
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode			
<b>MUL = multiply (unsigned)</b>								
accumulator with register/memory	<table><tr><td>1 1 1 1 0 1 1 w</td><td>mod 1 0 0 r/m</td></tr></table>	1 1 1 1 0 1 1 w	mod 1 0 0 r/m					
1 1 1 1 0 1 1 w	mod 1 0 0 r/m							
multiplier								
— byte		12-17/ 15-20*	12-17/ 15-20*	b, d	d, h			
— word		12-25/ 15-28*	12-25/ 15-28*	b, d	d, h			
— doubleword		12-41/ 17-46*	12-41/ 17-46*	b, d	d, h			
<b>IMUL = Integer multiply (signed)</b>								
accumulator with register/memory	<table><tr><td>1 1 1 1 0 1 1 w</td><td>mod 1 0 1 r/m</td></tr></table>	1 1 1 1 0 1 1 w	mod 1 0 1 r/m					
1 1 1 1 0 1 1 w	mod 1 0 1 r/m							
multiplier								
— byte		12-17/ 15-20*	12-17/ 15-20*	b, d	d, h			
— word		12-25/ 15-28*	12-25/ 15-28*	b, d	d, h			
— doubleword		12-41/ 17-46*	12-41/ 17-46*	b, d	d, h			
register with register/memory	<table><tr><td>0 0 0 0 1 1 1 1</td><td>1 0 1 0 1 1 1 1</td><td>mod reg r/m</td></tr></table>	0 0 0 0 1 1 1 1	1 0 1 0 1 1 1 1	mod reg r/m				
0 0 0 0 1 1 1 1	1 0 1 0 1 1 1 1	mod reg r/m						
multiplier								
— byte		12-17/ 15-20*	12-17/ 15-20*	b, d	d, h			
— word		12-25/ 15-28*	12-25/ 15-28*	b, d	d, h			
— doubleword		12-41/ 17-46*	12-41/ 17-46*	b, d	d, h			
register/memory with immediate to register	<table><tr><td>0 1 1 0 1 0 s 1</td><td>mod reg r/m</td></tr></table> immediate data	0 1 1 0 1 0 s 1	mod reg r/m					
0 1 1 0 1 0 s 1	mod reg r/m							
— word		13-26	13-26/ 14-27	b, d	d, h			
— doubleword		13-42	13-42/ 16-45	b, d	d, h			
<b>DIV = Divide (unsigned)</b>								
Accumulator by register/memory	<table><tr><td>1 1 1 1 0 1 1 w</td><td>mod 1 1 0 r/m</td></tr></table>	1 1 1 1 0 1 1 w	mod 1 1 0 r/m					
1 1 1 1 0 1 1 w	mod 1 1 0 r/m							
divisor								
— byte		14/17	14/17	b, e	e, h			
— word		22/25	22/25	b, e	e, h			
— doubleword		38/43	38/43	b, e	e, h			



## INSTRUCTION SET SUMMARY

Table E-1. Instruction Set Summary (Sheet 6 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Pro- tected Virtual Ad- dress Mode	Real Ad- dress Mode or Virtual 8086 Mode	Pro- tected Virtual Ad- dress Mode
IDIV = Integer divide (signed)					
Accumulator by register/memory	1 1 1 1 0 1 1 w    mod 111 r/m				
divisor					
— byte		19/22	19/22	b, e	e, h
— word		27/30	27/30	b, e	e, h
— doubleword		43/48	43/48	b, e	e, h
AAD = ASCII adjust for divide	1 1 0 1 0 1 0 1    0 0 0 0 1 0 1 0	19	19		
AAM = ASCII adjust for multiply	1 1 0 1 0 1 0 0    0 0 0 0 1 0 1 0	17	17		
CBW = Convert byte to word	1 0 0 1 1 0 0 0	3	3		
CWD = Convert word to double-word	1 0 0 1 1 0 0 1	2	2		
LOGIC					
shift rotate instructions not through carry (ROL, ROR, SAL, SAR, SHL, and SHR)					
register/memory by 1	1 1 0 1 0 0 0 w    mod TTT r/m	3/7**	3/7**	b	h
register/memory by CL	1 1 0 1 0 0 1 w    mod TTT r/m	3/7*	3/7*	b	h
register/memory by immediate count	1 1 0 0 0 0 0 w    mod TTT r/m	3/7*	3/7*	b	h
through carry (RCL and RCR)					
register/memory by 1	1 1 0 1 0 0 0 w    mod TTT r/m	9/10*	9/10*	b	h
register/memory by CL	1 1 0 1 0 0 1 w    mod TTT r/m	9/10*	9/10*	b	h
register/memory by immediate count	1 1 0 0 0 0 0 w    mod TTT r/m	9/10*	9/10*	b	h
T T T 0 0 0    Instruction 0 0 1    ROL 0 1 0    ROR 0 1 0    RCL 0 1 1    RCR 1 0 0    SHL/SAL 1 0 1    SHR 1 1 1    SAR					
SHLD = Shift left double					
register/memory by immediate	0 0 0 0 1 1 1 1    1 0 1 0 0 1 0 0    mod reg r/m	3/7**	3/7**		
register/memory by CL	0 0 0 0 1 1 1 1    1 0 1 0 0 1 0 1    mod reg r/m	3/7**	3/7**		



Table E-1. Instruction Set Summary (Sheet 7 of 19)

Instruction	Format	Clock Count		Notes					
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode				
<b>SHRD = Shift right double</b>									
register/memory by immediate	<table><tr><td>00001111</td><td>10101100</td><td>mod reg r/m</td></tr></table>	00001111	10101100	mod reg r/m	immed 8-bit data	3/7**	3/7**		
00001111	10101100	mod reg r/m							
register/memory by CL	<table><tr><td>00001111</td><td>10101101</td><td>mod reg r/m</td></tr></table>	00001111	10101101	mod reg r/m		3/7**	3/7**		
00001111	10101101	mod reg r/m							
<b>AND = And</b>									
register to register	<table><tr><td>001000dw</td><td>mod reg r/m</td></tr></table>	001000dw	mod reg r/m		2	2			
001000dw	mod reg r/m								
register to memory	<table><tr><td>0010000w</td><td>mod reg r/m</td></tr></table>	0010000w	mod reg r/m		7**	7**	b	h	
0010000w	mod reg r/m								
memory to register	<table><tr><td>0010001w</td><td>mod reg r/m</td></tr></table>	0010001w	mod reg r/m		6*	6*	b	h	
0010001w	mod reg r/m								
immediate to register/memory	<table><tr><td>1000000w</td><td>mod 100 r/m</td></tr></table>	1000000w	mod 100 r/m	immediate data	2/7*	2/7*	b	h	
1000000w	mod 100 r/m								
immediate to accumulator (short form)	<table><tr><td>0010010w</td><td>immediate data</td></tr></table>	0010010w	immediate data		2	2			
0010010w	immediate data								
<b>TEST= And function to flags, no result</b>									
register/memory and register	<table><tr><td>1000010w</td><td>mod reg r/m</td></tr></table>	1000010w	mod reg r/m		2/5*	2/5*	b	h	
1000010w	mod reg r/m								
immediate data and register/memory	<table><tr><td>1111011w</td><td>mod 000 r/m</td></tr></table>	1111011w	mod 000 r/m	immediate data	2/5*	2/5*	b	h	
1111011w	mod 000 r/m								
immediate data and accumulator (short form)	<table><tr><td>1010100w</td><td>immediate data</td></tr></table>	1010100w	immediate data		2	2			
1010100w	immediate data								
<b>OR = Or</b>									
register to register	<table><tr><td>000010dw</td><td>mod reg r/m</td></tr></table>	000010dw	mod reg r/m		2	2			
000010dw	mod reg r/m								
register to memory	<table><tr><td>0000100w</td><td>mod reg r/m</td></tr></table>	0000100w	mod reg r/m		7**	7**	b	h	
0000100w	mod reg r/m								
memory to register	<table><tr><td>0000101w</td><td>mod reg r/m</td></tr></table>	0000101w	mod reg r/m		6*	6*	b	h	
0000101w	mod reg r/m								
immediate to register/memory	<table><tr><td>1000000w</td><td>mod 001 r/m</td></tr></table>	1000000w	mod 001 r/m	immediate data	2/7**	2/7**	b	h	
1000000w	mod 001 r/m								
immediate to accumulator (short form)	<table><tr><td>0000110w</td><td>immediate data</td></tr></table>	0000110w	immediate data		2	2			
0000110w	immediate data								
<b>XOR = Exclusive or</b>									
register to register	<table><tr><td>001100dw</td><td>mod reg r/m</td></tr></table>	001100dw	mod reg r/m		2	2			
001100dw	mod reg r/m								
register to memory	<table><tr><td>0011000w</td><td>mod reg r/m</td></tr></table>	0011000w	mod reg r/m		7**	7**	b	h	
0011000w	mod reg r/m								
memory to register	<table><tr><td>0011001w</td><td>mod reg r/m</td></tr></table>	0011001w	mod reg r/m		6*	6*	b	h	
0011001w	mod reg r/m								
immediate to register/memory	<table><tr><td>1000000w</td><td>mod 110 r/m</td></tr></table>	1000000w	mod 110 r/m	immediate data	2/7**	2/7**	b	h	
1000000w	mod 110 r/m								
immediate to accumulator (short form)	<table><tr><td>0011010w</td><td>immediate data</td></tr></table>	0011010w	immediate data		2	2			
0011010w	immediate data								
<b>NOT= Invert register/memory</b>	<table><tr><td>1111011w</td><td>mod 010 r/m</td></tr></table>	1111011w	mod 010 r/m		2/6**	2/6**	b	h	
1111011w	mod 010 r/m								



## INSTRUCTION SET SUMMARY

Table E-1. Instruction Set Summary (Sheet 8 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
STRING MANIPULATION INSTRUCTIONS		Clk Count Virtual 8086 Mode			
CMPS = Compare byte word	1 0 1 0 0 1 1 w		10*	10*	b h
INS = Input byte/word from DX port	0 1 1 0 1 1 0 w	†30	17	10*/32**	b s/t, h, m
LODS = Load byte/word to AL/AX/EAX	1 0 1 0 1 1 0 w		5	5*	b h
MOVS = Move byte word	1 0 1 0 0 1 0 w		7	7**	b h
OUTS = Output byte/word to DX port	0 1 1 0 1 1 1 w	†31	18	11*/33*	b s/t, h, m
SCAS = Scan byte word	1 0 1 0 1 1 1 w		7*	7*	b h
STOS = Store byte/word from AL/AX/EX	1 0 1 0 1 0 1 w		4*	4*	b h
XLAT = Translate String	1 1 0 1 0 1 1 1		5*	5*	h
REPEATED STRING MANIPULATION Repeated by count in CX or ECX:					
REPE CMPS = Compare string					
find non-match	1 1 1 1 0 0 1 1 1 0 1 0 0 1 1 w	Clk Count Virtual 8086 Mode	5 + 9n**	5 + 9n**	b h
find match	1 1 1 1 0 0 1 0 1 0 1 0 0 1 1 w		5 + 9n**	5 + 9n**	b h
REP INS = Input string	1 1 1 1 0 0 1 0 0 1 1 0 1 1 0 w	†31+6n	17 + 7n*	11 + 7n*/32+ 6n*	b s/t, h, m
REP LODS = Load string	1 1 1 1 0 0 1 0 1 0 1 0 1 1 0 w		5 + 6n*	5 + 6n*	b h
REP MOVS = Move string	1 1 1 1 0 0 1 0 1 0 1 0 0 1 0 w		7 + 4n*	7 + 4n**	b h
REP OUTS = Output string	1 1 1 1 0 0 1 0 0 1 1 0 1 1 1 w	†30+8n	16 + 8n*	10 + 8n*/31+ 8n*	b s/t, h, m
REPE SCAS = Scan string Find non-AL/AX/EAX	1 1 1 1 0 0 1 1 1 0 1 0 1 1 1 w		5 + 8n*	5 + 8n*	b h
REPNE SCAS = Scan string Find AL/AX/EAX	1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 w		5 + 8n*	5 + 8n*	b h
REP STOS = Store string	1 1 1 1 0 0 1 0 1 0 1 0 1 0 1 w		5 + 5n*	5 + 5n*	b h
BIT MANIPULATION					
BSF = scan bit forward	0 0 0 0 1 1 1 1 1 0 1 1 1 1 0 0 mod reg r/m		10 + 3n*	10 + 3n*	b h
BSR = scan bit reverse	0 0 0 0 1 1 1 1 1 0 1 1 1 1 0 1 mod reg r/m		10 + 3n*	10 + 3n*	b h
BT = test bit					
register/memory, immediate	0 0 0 0 1 1 1 1 1 0 1 1 1 1 0 1 0 mod 1 0 0 r/m	immed 8-bit data	3/6*	3/6*	b h



Table E-1. Instruction Set Summary (Sheet 9 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
register/memory, register	00001111 10100011 mod reg r/m	3/12*	3/12*	b	h
<b>BTC = test bit and complement</b>					
register/memory, immediate	00001111 10111010 mod 111 r/m	6/8*	6/8*	b	h
register/memory, register	00001111 10111011 mod reg r/m	6/13*	6/13*	b	h
<b>BTR = test bit and reset</b>					
register/memory, immediate	00001111 10111010 mod 110 r/m	6/8*	6/8*	b	h
register/memory, register	00001111 10110011 mod reg r/m	6/13*	6/13*	b	h
<b>BTS = test bit and set</b>					
register/memory, immediate	00001111 10111010 mod 101 r/m	6/8*	6/8*	b	h
register/memory, register	00001111 10101011 mod reg r/m	6/13*	6/13*	b	h
<b>CONTROL TRANSFER</b>					
<b>CALL = Call</b>					
direct within segment	11101000 full displacement	7 + m*	9 + m*	b	r
reg/memory indirect within segment	11111111 mod 010 r/m	7 + m*/ 10 + m*	9 + m*/ 12 + m*	b	h, r
direct intersegment	10011010 unsigned full offset, selector	17 + m*	42 + m*	b	j, k, r
Protected mode only (direct intersegment)					
Via call gate to same privilege level			64 + m		h, j, k, r
Via call gate to different privilege level (no parameters)			98 + m		h, j, k, r
Via call gate to different privilege level (x parameters)			106+8x+m		h, j, k, r
From 286 task to 286 TSS			285		h, j, k, r
From 286 task to Intel386 SX CPU TSS			310		h, j, k, r
From 286 task to virtual 8086 task (Intel386 SX CPU TSS)			229		h, j, k, r
From Intel386 SX CPU task to 286 TSS			285		h, j, k, r
From Intel386 SX CPU task to Intel386 SX CPU TSS			392		h, j, k, r
From Intel386 SX CPU task to Virtual 8086 task (Intel386 SX CPU TSS)			309		h, j, k, r
indirect intersegment	11111111 mod 011 r/m	30 + m	46 + m	b	h, j, k, r
Protected mode only (indirect intersegment)					
Via call gate to same privilege level			68 + m		h, j, k, r



## INSTRUCTION SET SUMMARY

Table E-1. Instruction Set Summary (Sheet 10 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
Via call gate to different privilege level (no parameters)			102 + m		h, j, k, r
Via call gate to different privilege level (x parameters)			110+8x+m		h, j, k, r
From 286 task to 286 TSS					h, j, k, r
From 286 task to Intel386 SX CPU TSS					h, j, k, r
From 286 task to virtual 8086 task (Intel386 SX CPU TSS)					h, j, k, r
From Intel386 SX CPU task to 286 TSS					h, j, k, r
From Intel386 SX CPU task to Intel386 SX CPU TSS			399		h, j, k, r
From Intel386 SX CPU task to Virtual 8086 task (Intel386 SX CPU TSS)					h, j, k, r
JMP = Unconditional jump					
short	1 1 1 0 1 0 1 1    8-bit displacement	7 + m	7 + m		r
direct within segment	1 1 1 0 1 0 0 1    full displacement	7 + m	7 + m		r
reg/memory indirect within segment	1 1 1 1 1 1 1 1    mod 1 0 0 r/m	9 + m/ 14 + m	9 + m/ 14 + m	b	h, r
direct intersegment	1 1 1 0 1 0 1 0    unsigned full offset, selector	16 + m	31 + m		j, k, r
Protected mode only (direct intersegment)					
Via call gate to same privilege level			53 + m		h, j, k, r
From 286 task to 286 TSS					h, j, k, r
From 286 task to Intel386 SX CPU TSS					h, j, k, r
From 286 task to virtual 8086 task (Intel386 SX CPU TSS)					h, j, k, r
From Intel386 SX CPU task to 286 TSS					h, j, k, r
From Intel386 SX CPU task to Intel386 SX CPU TSS					h, j, k, r
From Intel386 SX CPU task to Virtual 8086 task (Intel386 SX CPU TSS)			395		h, j, k, r
indirect intersegment	1 1 1 1 1 1 1 1    mod 1 0 1 r/m	17 + m	31 + m	b	h, j, k, r
Protected mode only (indirect intersegment)					
Via call gate to same privilege level			49 + m		h, j, k, r
From 286 task to 286 TSS					h, j, k, r
From 286 task to Intel386 SX CPU TSS					h, j, k, r
From 286 task to virtual 8086 task (Intel386 SX CPU TSS)					h, j, k, r
From Intel386 SX CPU task to 286 TSS					h, j, k, r
From Intel386 SX CPU task to Intel386 SX CPU TSS			328		h, j, k, r
From Intel386 SX CPU task to Virtual 8086 task (Intel386 SX CPU TSS)					h, j, k, r
RET = Return from CALL					
within segment	1 1 0 0 0 0 1 1		12 + m	b	g, h, r



Table E-1. Instruction Set Summary (Sheet 11 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
within segment adding immed to SP	11000010 16-bit displacement		12 + m	b	g, h, r
intersegment	11001011		36 + m	b	g, h, j, k, r
intersegment adding immed to SP	11001010 16-bit displacement		36 + m	b	g, h, j, k, r
Protected mode only (RET):					
to different privilege level					
Intersegment			72		h, j, k, r
Intersegment adding immed to SP			72		h, j, k, r
<b>CONDITIONAL JUMPS</b> (times are jump "Taken or not Taken")					
JO = jump on overflow					
8-bit displacement	01110000 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10000000 Full displacement	7 + m or 3	7 + m or 3		r
JNO = Jump on not overflow					
8-bit displacement	01110001 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10000001 Full displacement	7 + m or 3	7 + m or 3		r
JB/JNAE = jump on below/not above or equal					
8-bit displacement	01110010 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10000010 Full displacement	7 + m or 3	7 + m or 3		r
JNB/JAE = jump on not below/above or equal					
8-bit displacement	01110011 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10000011 Full displacement	7 + m or 3	7 + m or 3		r
JE/JZ= jump on equal/zero					
8-bit displacement	01110100 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10000100 Full displacement	7 + m or 3	7 + m or 3		r
JNE/JNZ = jump on not equal/not zero					
8-bit displacement	01110101 8-bit displacement	7 + m or 3	7 + m or 3		r





## INSTRUCTION SET SUMMARY

Table E-1. Instruction Set Summary (Sheet 12 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
Full displacement	00001111 10000101 Full displacement	7 + m or 3	7 + m or 3		r
JBE/JNA = jump on below or equal/not above					
8-bit displacement	01110110 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10000110 Full displacement	7 + m or 3	7 + m or 3		r
JNBE/JA = jump on not below or equal/above					
8-bit displacement	01110111 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10000111 Full displacement	7 + m or 3	7 + m or 3		r
JS = jump on sign					
8-bit displacement	01111000 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10001000 Full displacement	7 + m or 3	7 + m or 3		r
JNS = jump on not sign					
8-bit displacement	01111001 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10001001 Full displacement	7 + m or 3	7 + m or 3		r
JPI/JPE = jump on parity/parity even					
8-bit displacement	01111010 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10001010 Full displacement	7 + m or 3	7 + m or 3		r
JNP/JPO = jump on not parity/parity odd					
8-bit displacement	01111011 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10001011 Full displacement	7 + m or 3	7 + m or 3		r
JL/JNGE = jump on less/not greater or equal					
8-bit displacement	01111100 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10001100 Full displacement	7 + m or 3	7 + m or 3		r
JNL/JGE = jump on not less/greater or equal					
8-bit displacement	01111101 8-bit displacement	7 + m or 3	7 + m or 3		r



Table E-1. Instruction Set Summary (Sheet 13 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
Full displacement	00001111 10001101 Full displacement	7 + m or 3	7 + m or 3		r
JLE/JNG = jump on less or equal/not greater					
8-bit displacement	01111110 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10001110 Full displacement	7 + m or 3	7 + m or 3		r
JNLE/JG = jump on not less or equal/greater					
8-bit displacement	01111111 8-bit displacement	7 + m or 3	7 + m or 3		r
Full displacement	00001111 10001111 Full displacement	7 + m or 3	7 + m or 3		r
JCXZ = jump on CX zero	11100011 8-bit displacement	9 + m or 5	9 + m or 5		r
JECXZ = jump on ECX zero	11100011 8-bit displacement	9 + m or 5	9 + m or 5		r
(Address size prefix differentiates JCXZ from JECXZ)					
LOOP = loop CX times	11100010 8-bit displacement	11 + m	11 + m		r
LOOPZ/LOOPE = loop with zero/equal	11100001 8-bit displacement	11 + m	11 + m		r
LOOPNZ/LOOPNE = loop while not zero	11100000 8-bit displacement	11 + m	11 + m		r
CONDITIONAL BYTE SET					
(Note: Times are register/memory)					
SETO = set byte on overflow					
to register/memory	00001111 10010000 mod 000 r/m	4/5*	4/5*		h
SETNO = set byte on not overflow					
to register/memory	00001111 10010001 mod 000 r/m	4/5*	4/5*		h
SETB/SETNAE = set byte on below/not above or equal					
to register/memory	00001111 10010010 mod 000 r/m	4/5*	4/5*		h
SETNB = set byte on below/above or equal					
to register/memory	00001111 10010011 mod 000 r/m	4/5*	4/5*		h
SETE/SETZ = set byte on equal/zero					
to register/memory	00001111 10010100 mod 000 r/m	4/5*	4/5*		h
SETNE/SETNZ = set byte on not equal/not zero					
to register/memory	00001111 10010101 mod 000 r/m	4/5*	4/5*		h



## INSTRUCTION SET SUMMARY

Table E-1. Instruction Set Summary (Sheet 14 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
<b>SETBE/SETNA</b> = set byte on below or equal/not above					
to register/memory	00001111 10010110 mod 000 r/m	4/5*	4/5*		h
<b>SETNBE/SETA</b> = set byte on not below or equal/above					
to register/memory	00001111 10010111 mod 000 r/m	4/5*	4/5*		h
<b>SETS</b> = set byte on sign					
to register/memory	00001111 10011000 mod 000 r/m	4/5*	4/5*		h
<b>SETNS</b> = set byte on not sign					
to register/memory	00001111 10011001 mod 000 r/m	4/5*	4/5*		h
<b>SETP/SETPE</b> = set byte on parity/parity even					
to register/memory	00001111 10011010 mod 000 r/m	4/5*	4/5*		h
<b>SETNP/SETPO</b> = set byte on not parity/parity odd					
to register/memory	00001111 10011011 mod 000 r/m	4/5*	4/5*		h
<b>SETL/SETNGE</b> = set byte on less/not greater or equal					
to register/memory	00001111 10011100 mod 000 r/m	4/5*	4/5*		h
<b>SETNL/SETGE</b> = set byte on not less/greater or equal					
to register/memory	00001111 01111101 mod 000 r/m	4/5*	4/5*		h
<b>SETLE/SETNG</b> = set byte on less or equal/not greater					
to register/memory	00001111 10011110 mod 000 r/m	4/5*	4/5*		h
<b>SETNLE/SETG</b> = set byte on not less or equal/greater					
to register/memory	00001111 10011111 mod 000 r/m	4/5*	4/5*		h
<b>ENTER</b> = enter procedure	11001000 16-bit displacement, 8-bit level				
L = 0		10	10	b	h
L = 1		14	14	b	h
L > 1		17+8(n-1)	17+8(n-1)		
<b>LEAVE</b> = leave procedure	11001001	4	4	b	h
<b>INTERRUPT INSTRUCTIONS</b>					
<b>INT</b> = Interrupt:					
Type specified	11001101 type	37		b	
Type 3	11001100	33		b	
<b>INTO</b> = Interrupt 4 if overflow flag set	11001110				
If OF = 1		35		b, e	
If OF = 0		3	3	b, e	



Table E-1. Instruction Set Summary (Sheet 15 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
<b>BOUND</b> = Interrupt 5 if Detect value out of range If out of range If in range Protected Mode Only (INT) INT: Type Specified Via interrupt or Trap Gate To same privilege level Via Interrupt or Trap Gate To different privilege level From 286 task to 286 TSS via Task Gate From 286 task to Intel 386 SX CPU TSS via Task Gate From 286 task to virtual 8086 mode via Task Gate From Intel386 SX CPU task to 286 TSS via Task Gate From Intel386 SX CPU task to Intel 386 SX CPU TSS via task gate From Intel386 SX CPU task to virtual 8086 mode via Task Gate From virtual 8086 mode to 286 TSS via Task Gate From virtual 8086 mode to Intel 386 SX CPU TSS via task gate From virtual 8086 mode to privilege level 0 via trap gate or interrupt gate INT: TYPE 3 Via interrupt or Trap Gate To same privilege level Via Interrupt or Trap Gate To different privilege level From 286 task to 286 TSS via Task Gate From 286 task to Intel 386 SX CPU TSS via Task Gate From 286 task to virtual 8086 mode via Task Gate From Intel386 SX CPU task to 286 TSS via Task Gate From Intel386 SX CPU task to Intel 386 SX CPU TSS via task gate From Intel386 SX CPU task to virtual 8086 mode via Task Gate From virtual 8086 mode to 286 TSS via Task Gate From virtual 8086 mode to Intel 386 SX CPU TSS via task gate From virtual 8086 mode to privilege level 0 via trap gate or interrupt gate	<div>0 1 1 0 0 0 1 0</div> <div>mod reg r/m</div>	44		b, e	e, g, h, j, k, r
		10	10	b, e	e, g, h, j, k, r
			71		g, j, k, r
			111		g, j, k, r
			438		g, j, k, r
			465		g, j, k, r
			382		g, j, k, r
			440		g, j, k, r
			467		g, j, k, r
			384		g, j, k, r
			445		g, j, k, r
			472		g, j, k, r
			275		g, j, k, r
			71		g, j, k, r
			111		g, j, k, r
			382		g, j, k, r
			409		g, j, k, r
			326		g, j, k, r
			384		g, j, k, r
			411		g, j, k, r
			328		g, j, k, r
			389		g, j, k, r
			416		g, j, k, r
			223		





## INSTRUCTION SET SUMMARY

Table E-1. Instruction Set Summary (Sheet 16 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
INTO:					
Via interrupt or Trap Gate To same privilege level			71		g, j, k, r
Via Interrupt or Trap Gate To different privilege level			111		g, j, k, r
From 286 task to 286 TSS via Task Gate			384		g, j, k, r
From 286 task to Intel 386 SX CPU TSS via Task Gate			411		g, j, k, r
From 286 task to virtual 8086 mode via Task Gate			328		g, j, k, r
From Intel386 SX CPU task to 286 TSS via Task Gate			Intel386 DX		g, j, k, r
From Intel386 SX CPU task to Intel 386 SX CPU TSS via task gate			413		g, j, k, r
From Intel386 SX CPU task to virtual 8086 mode via Task Gate			329		g, j, k, r
From virtual 8086 mode to 286 TSS via Task Gate			391		g, j, k, r
From virtual 8086 mode to Intel 386 SX CPU TSS via task gate			418		g, j, k, r
From virtual 8086 mode to privilege level 0 via trap gate or interrupt gate			223		
BOUND:					
Via interrupt or Trap Gate To same privilege level			71		g, j, k, r
Via Interrupt or Trap Gate To different privilege level			111		g, j, k, r
From 286 task to 286 TSS via Task Gate			358		g, j, k, r
From 286 task to Intel 386 SX CPU TSS via Task Gate			388		g, j, k, r
From 286 task to virtual 8086 mode via Task Gate			335		g, j, k, r
From Intel386 SX CPU task to 286 TSS via Task Gate			368		g, j, k, r
From Intel386 SX CPU task to Intel 386 SX CPU TSS via task gate			398		g, j, k, r
From Intel386 SX CPU task to virtual 8086 mode via Task Gate			347		g, j, k, r
From virtual 8086 mode to 286 TSS via Task Gate			368		g, j, k, r
From virtual 8086 mode to Intel 386 SX CPU TSS via task gate			398		g, j, k, r
From virtual 8086 mode to privilege level 0 via trap gate or interrupt gate			223		
INTERRUPT RETURN					
IRET = Interrupt return	1 1 0 0 1 1 1 1	24			g, h, j, k, r
Protected Mode Only (IRET)					
To same privilege level (within task)			42		g, h, j, k, r



Table E-1. Instruction Set Summary (Sheet 17 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
To different privilege level (within task)			86		g, h, j, k, r
From 286 task to 286 TSS			285		g, h, j, k, r
From 286 task to Intel386 SX CPU TSS			318		g, h, j, k, r
From 286 task to virtual 8086 task			267		g, h, j, k, r
From 286 task to virtual 8086 mode (within task)			113		
From Intel386 SX CPU task to 286 TSS			324		g, h, j, k, r
From Intel386 SX CPU task to Intel386 SX CPU TSS			328		g, h, j, k, r
From Intel386 SX CPU task to virtual 8086 task			377		g, h, j, k, r
From Intel386 SX CPU task to virtual 8086 mode (within task)			113		
PROCESSOR CONTROL INSTRUCTIONS					
HLT = Halt	11110100	7	7		l
MOV = Move to and from control/debug/test registers					
CR0/CR2/CR3 from register	00001111 00100010 11eee reg	10/4/5	10/4/5		l
register from CR0-3	00001111 00100000 11eee reg	6	6		l
DR0-3 from register	00001111 00100011 11eee reg	22	22		l
DR6-7 from register	00001111 00100011 11eee reg	16	16		l
register from DR6-7	00001111 00100001 11eee reg	14	14		l
register from DR0-3	00001111 00100001 11eee reg	22	22		l
TR6-7 from register	00001111 00100110 11eee reg	12	12		l
register from TR6-7	00001111 00100100 11eee reg	12	12		l
NOP = No operation	10010000	3	3		
WAIT = Wait until BUSY# pin is negated	10011011	6	6		
PROCESSOR EXTENSION INSTRUCTIONS					
Processor Extension Escape	11011TTT mod LLL r/m	See Intel387 SX datasheet for clock counts			h
TTT and LLL bits are opcode information for coprocessor					



## INSTRUCTION SET SUMMARY

Table E-1. Instruction Set Summary (Sheet 18 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
PREFIX BYTES					
Address size prefix	01100111	0	0		
LOCK = Bus lock prefix	11110000	0	0		m
Operand size prefix	01100110	0	0		
Segment override prefix					
CS:	00101110	0	0		
DS:	00111110	0	0		
ES:	00100110	0	0		
FS:	01100100	0	0		
GS:	01100101	0	0		
SS:	00110110	0	0		
PROTECTION CONTROL					
ARPL = adjust requested privilege level					
from register/memory	01100011 mod reg r/m	N/A	20/21**	a	h
LAR = load access rights					
from register/memory	00001111 00000010 mod reg r/m	N/A	15/16*	a	g, h, j, p
LGDT = load global descriptor					
table register	00001111 00000001 mod 010 r/m	11*	11*	b, c	h, l
LIDT = load interrupt descriptor					
table register	00001111 00000001 mod 011 r/m	11*	11*	b, c	h, l
LLDT = load local descriptor					
table register to register/memory	00001111 00000000 mod 010 r/m	N/A	20/24*	a	g, h, j, l
LMSW = load machine status word					
from register/memory	00001111 00000001 mod 110 r/m	10/13	10/13*	b, c	h, l
LSL = load segment limit					
from register memory	00001111 00000011 mod reg r/m				
Byte-Granular limit		N/A	20/21*	a	g, h, j, p
Page-Granular limit		N/A	25/26*	a	g, h, j, p
LTR = load task register					
from register/memory	00001111 00000000 mod 001 r/m	N/A	23/27*	a	g, h, j, l
SGDT = store global descriptor					
table register	00001111 00000001 mod 000 r/m	9*	9*	b, c	h
SIDT = store interrupt descriptor					



Table E-1. Instruction Set Summary (Sheet 19 of 19)

Instruction	Format	Clock Count		Notes	
		Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode	Real Address Mode or Virtual 8086 Mode	Protected Virtual Address Mode
table register	00001111 00000001 mod 001 r/m	9*	9*	b, c	h
SLDT = store local descriptor table					
to register/memory	00001111 00000000 mod 000 r/m	N/A	2/2*	a	h
SMSW = store machine status word	00001111 00000001 mod 100 r/m	2/2*	2/2*	b, c	h, i
STR = store task register					
to register/memory	00001111 00000000 mod 001 r/m	N/A	2/2*	a	h
VERR = verify read access					
register/memory	00001111 00000000 mod 100 r/m	N/A	10/11*	a	g, h, j, p
VERW = verify write access	00001111 00000000 mod 101 r/m	N/A	15/16*	a	g, h, j, p

**NOTES:****Notes a through c apply to Real Address Mode only:**

- This is a Protected Mode instruction. Attempted execution in Real Mode will result in exception 6 (invalid opcode).
- Exception 13 fault (general protection) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum CS, DS, ES, FS, or GS limit, FFFFH. Exception 12 fault (stack segment limit violation or not present) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum SS limit.
- This instruction may be executed in Real Mode, its purpose is primarily to initialize the CPU for Protected Mode.

**Notes d through g apply to Real Address Mode and Protected Virtual Address Mode:**

- The Intel386 SX CPU uses an early-out multiply algorithm. The actual number of clocks depends on the position of the most significant bit in the operand (multiplier).

Clock counts given are minimum to maximum. To calculate actual clocks use this formula:

Actual Clock = if  $m < 0$  then  $\max([\log_2 |m|], 3) + b$  clocks:  
if  $m = 0$  then  $3 + b$  clocks

In this formula, m is the multiplier, and

b = 9 for register to register  
b = 12 for memory to register





## INSTRUCTION SET SUMMARY

b = 10 for register with immediate to register  
b = 11 for memory with immediate to register.

- e. An exception may occur, depending on the value of the operand.
- f. LOCK# is automatically asserted, regardless of the presence or absence of the LOCK# prefix.
- g. LOCK# is asserted during descriptor table accesses.

**Notes h through r apply to Protected Virtual Address Mode only:**

- h. Exception 13 fault (general protection violation) will occur if the memory operand in CS, DS, ES, FS, or GS cannot be used due to either a segment limit violation or access rights violation. If a stack limit is violated, an exception 12 (stack segment limit violation or not present) occurs.
- i. For segment load operations, the CPL, RPL, and DPL must agree with the privilege rules to avoid an exception 13 fault (general protection violation). The segment's descriptor must indicate "present" or exception 11 (CS, DS, ES, FS, GS not present.) If the SS register is loaded and a stack segment not present is detected, an exception 12 (stack segment limit violation or not present) occurs.
- j. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert LOCK# to maintain descriptor integrity in multiprocessor systems.
- k. JMP, CALL, INT, RET, and IRET instructions referring to another code segment will cause an exception 13 (general protection violation) if an applicable privilege rule is violated.
- l. An exception 13 fault occurs if CPL is greater than 0 (0 is the most privileged level)
- m. An exception 13 fault occurs if CPL is greater than IOPL.
- n. The IF bit of the flag register is not updated if CPL is greater than IOPL. The IOPL and VM fields of the flag register are updated only if CPL = 0.
- o. The PE bit of the MSW (CR0) cannot be reset by this instruction. Use MOV into CR0 when resetting the PE bit.
- p. Any violation of privilege rules as applied to the selector operand does not cause a protection exception; rather the zero flag is cleared.
- q. If the coprocessor's memory operand violates a segment limit or segment access rights, an exception 13 fault (general protection exception) will occur before the ESC instruction is executed. An exception 12 fault (stack segment limit violation or not present) will occur if the stack limit is violated by the operand's starting address.
- r. The destination of a JMP, CALL, INT, RET, or IRET must be in the defined limit of a code segment or an exception 13 fault (general protection violation) will occur.
- s. The instruction will execute in s clocks if  $CPL \leq IOPL$ . If  $CPL > IOPL$ , the instruction will take t clocks.
- † Clock count shown applies if I/O permission allows I/O to the port in virtual 8086 mode. If I/O bit map denies permission, exception 13 (general protection fault) occurs; refer to clock counts for INT 3 instruction.



## E.2 INSTRUCTION ENCODING

All instruction encodings are subsets of the general instruction format shown in Figure E-1. Instructions consist of one or two primary opcode bytes, possibly an address specifier consisting of the “mod r/m” byte and “scaled index” byte, a displacement if required, and an immediate data field if required.

Within the primary opcode or opcodes, smaller encoding fields may be defined. These fields vary according to the class of operation. The fields define such information as direction of the operation, size of the displacements, register encoding, or sign extension.

Almost all instructions referring to an operand in memory have an addressing mode byte following the primary opcode byte(s). This byte, the mod r/m byte, specifies the address mode to be used. Certain encodings of the mod r/m byte indicate a second addressing byte, the scale-index-base byte, which follows the mod r/m byte to fully specify the addressing mode.

Addressing modes can include a displacement immediately following the mod r/m byte, or scaled index byte. If a displacement is present, the possible sizes are 8, 16, or 32 bits.

If the instruction specifies an immediate operand, the immediate operand follows any displacement bytes. The immediate operand, if specified, is always the last field of the instruction.

Figure E-1 illustrates several of the fields that can appear in an instruction, such as the mod field and the r/m field, but the figure does not show all fields. Several smaller fields also appear in certain instructions, sometimes within the opcode bytes themselves. Table E-2 is a complete list of all fields appearing in the instruction set. Following Table E-2 are detailed tables for each field.

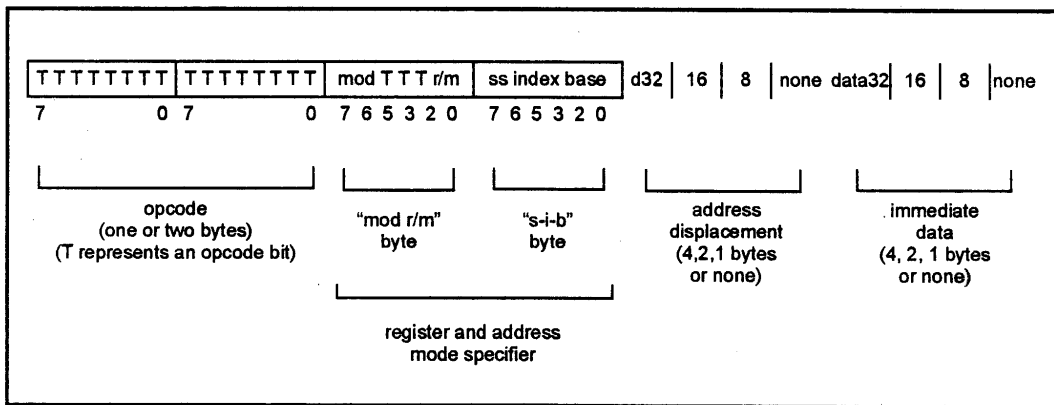


Figure E-1. General Instruction Format



## INSTRUCTION SET SUMMARY

Table E-2. Fields Within Instructions

Field Name	Description	Number of Bits
w	Specifies if data is byte of full size (full size is either 16 or 32 bits)	1
d	Specifies Direction of Data Operation	1
s	Specifies if an Immediate Data Field must be Sign-Extended	1
reg	General Register Specifier	3
mod r/m	Address Mode Specifier (Effective Address can be a General Register)	2 for mod: 3 for r/m
ss	Scale Factor for Scaled Index Address Mode	2
index	General Register to be used as Index Register	3
base	General Register to be used as Base Register	3
sreg2	Segment Register Specifier for CS, SS, DS, ES	2
sreg3	Segment Register Specifier for CS, SS, DS, ES, FS, GS	3
tttn	For Conditional Instructions, specifies a condition asserted or a condition negated	4

**NOTE:** Figure E-1 shows encoding of individual instructions.

### E.2.1 32-bit Extensions of the Instruction Set

With the Intel386 EX processor the 8086/80186/80286 instruction set is extended in two orthogonal directions: 32-bit forms of all 16-bit instructions are added to support the 32-bit data types, and 32-bit addressing modes are made available for all instructions referencing memory. This orthogonal instruction set extension is accomplished having a Default (D) bit in the code segment descriptor, and by having 2 prefixes to the instruction set.

The instruction defaults to operations of 16 bits or 32 bits, depending on the setting of the D bit in the code segment descriptor, which gives the default length (either 32 bites or 16 bits) for both operands and effective addresses when executing that code segment. In the Real Address Mode or Virtual 8086 Mode, no code segment descriptors are used, but a D value of 0 is assumed internally by the Intel386 EX processor when operating in those modes (for 16-bit default sizes compatible with the 8086/80186/80286).

Two prefixes, the Operand Size Prefix and the Effective Address Size Prefix, allow overriding individually the Default selection of operand size and effective address size. These prefixes may precede any opcode bytes and affect only the instruction they precede. If necessary, one or both prefixes may be placed before the opcode bytes. The presence of the Operand Size Prefix and the Effective Address Prefix toggles the operand size or the effective address size, respectively, to the value "opposite" from the default setting. For example, if the default operand size is for 32-bit data operations, then presence of the Operand Size Prefix toggles the instruction to use 32-bit effective address computations.

These 32-bit extensions are available in all modes, including the Real Address Mode or the Virtual 8086 Mode. In these modes the default is always 16 bits, so prefixes are needed to specify 32-bit operands or addresses. For instructions with more than one prefix the order of prefixes is unimportant.



Unless specified otherwise, instructions with 8-bit and 16-bit operands do not affect the contents of the high-order bits of the extended registers.

## E.2.2 Encoding of Instruction Fields

Within the instruction are several fields indicating register selection, addressing mode, and so on. The exact encodings of these fields are defined in the next several section.

### E.2.2.1 Encoding of Operand Length (w) Field

For any given instruction performing a data operation, the instruction is executing as a 32-bit operation or a 16-bit operation. Within the constraints of the operation size, the w field encodes the operand size as either one byte or the full operation size, as shown in Table E-3.

**Table E-3. Encoding of Operand Length (w) Field**

w Field	Operand Size During 16-bit Data Operations	Operand Size During 32-bit Data Operations
0	8 bits	8 bits
1	16 bits	32 bits

### E.2.2.2 Encoding of the General Register (reg) Field

The general register is specified by the reg field, which may appear in the primary opcode bytes, or as the reg field of the "mod r/m" byte, or as the r/m field of the "mod r/m" byte.

**Table E-4. Encoding of reg Field When w Field is not Present in Instruction**

reg Field	Register Selected During 16-bit Data Operations	Register Selected During 32-bit Data Operations
000	AX	EAX
001	CX	ECX
010	DX	EDX
011	BX	EBX
100	SP	ESP
101	BP	EBP
110	SI	ESI
111	DI	EDI



## INSTRUCTION SET SUMMARY

Table E-5. Encoding of reg Field When w Field is Present in Instruction

Register Specified by reg Field During 16-bit Data Operations		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

Register Specified by reg Field During 32-bit Data Operations		
reg	Function of w Field	
	(when w = 0)	(when w = 1)
000	AL	EAX
001	CL	ECX
010	DL	EDX
011	BL	EBX
100	AH	ESP
101	CH	EBP
110	DH	ESI
111	BH	EDI

## E.2.2.3 Encoding of the Segment Register (sreg) Field

The sreg field in certain instructions is a 2-bit field allowing one of the four 80286 segment registers to be specified. The sreg field in other instructions is a 3-bit field, allowing the FS and GS segment registers to be specified.

Table E-6. Encoding of the Segment Register (sreg) Field

2-Bit sreg2 Field	Segment Register Selected	3-Bit sreg3 Field	Segment Register Selected
00	ES	000	ES
01	CS	001	CS
10	SS	010	SS
11	DS	011	DS
		100	FS
		101	GS
		110	do not use
		111	do not use



#### E.2.2.4 Encoding of Address Mode

Except for special instructions, such as PUSH or POP, where the addressing mode is pre-determined, the addressing mode for the current instruction is specified by addressing bytes following the primary opcode. The primary addressing byte is the "mod r/m" byte, and a second byte of addressing information, the "s-i-b" (scale-index-base) byte can be specified.

The s-i-b byte is specified when using 32-bit addressing mode and the "mod r/m" byte has r/m = 100 and mod = 00, 01, or 10. When the s-i-b byte is present, the 32-bit addressing mode is a function of the mod, ss, index, and base fields.

The primary addressing byte, the "mod r/m" byte, also contains three bits (shown as TTT in Figure E-1) sometimes used as an extension of the primary opcode. The three bits, however, may also be used as a register field (reg).

When calculating an effective address, either 16-bit addressing or 32-bit addressing is used. 16-bit addressing uses 16-bit address components to calculate the effective address while 32-bit addressing uses 32-bit address components to calculate the effective address. When 16-bit addressing is used, the "mod r/m" byte is interpreted as a 32-bit addressing mode specifier.

The following tables define all encodings of all 16-bit addressing modes and 32-bit addressing modes.





## INSTRUCTION SET SUMMARY

Table E-7. Encoding of 16-bit Address Mode with "mod r/m" Byte

mod r/m	Effective Address	mod r/m	Effective Address
00 000	DS:[BX + SI]	10 000	DS:[BX + SI + d16]
00 001	DS:[BX + DI]	10 001	DS:[BX + DI + d16]
00 010	SS:[BP + SI]	10 010	SS:[BP + SI + d16]
00 011	SS:[BP + DI]	10 011	SS:[BX + DI + d16]
00 100	DS:[SI]	10 100	DS:[SI + d16]
00 101	DS:[DI]	10 101	DS:[DI + d16]
00 110	DS:d16	10 110	SS:[BP + d16]
00 111	DS:[BX]	10 111	DS:[BX + d16]
01 000	DS:[BX + SI + d8]	11 000	register - see tables below
01 001	DS:[BX + DI + d8]	11 001	register - see tables below
01 010	SS:[BP + SI + d8]	11 010	register - see tables below
01 011	SS:[BP + DI + d8]	11 011	register - see tables below
01 100	DS:[SI + d8]	11 100	register - see tables below
01 101	DS:[DI + d8]	11 101	register - see tables below
01 110	SS:[BP + d8]	11 110	register - see tables below
01 111	DS:[BX + d8]	11 111	register - see tables below

Register Specified by r/m During 16-bit Data Operations		
mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by r/m During 32-bit Data Operations		
mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI



Table E-8. Encoding of 32-bit Address Mode with “mod r/m” Byte (No s-i-b Byte Present)

mod r/m	Effective Address	mod r/m	Effective Address
00 000	DS:[EAX]	10 000	DS:[EAX + d32]
00 001	DS:[ECX]	10 001	DS:[ECX + d32]
00 010	SS:[EDX]	10 010	SS:[EDX + d32]
00 011	SS:[EBX]	10 011	SS:[EBX + d32]
00 100	s-i-b is present	10 100	s-i-b is present
00 101	DS:[d32]	10 101	SS:[EBP + d32]
00 110	DS:[ESI]	10 110	SS:[ESI + d32]
00 111	DS:[EDI]	10 111	DS:[EDI + d32]
01 000	DS:[EAX + d8]	11 000	register - see tables below
01 001	DS:[ECX + d8]	11 001	register - see tables below
01 010	SS:[EDX + d8]	11 010	register - see tables below
01 011	SS:[EBX + d8]	11 011	register - see tables below
01 100	s-i-b is present	11 100	register - see tables below
01 101	SS:[EBP + d8]	11 101	register - see tables below
01 110	DS:[ESI + d8]	11 110	register - see tables below
01 111	DS:[EDI + d8]	11 111	register - see tables below

Register Specified by r/m During 16-bit Data Operations		
mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	AX
11 001	CL	CX
11 010	DL	DX
11 011	BL	BX
11 100	AH	SP
11 101	CH	BP
11 110	DH	SI
11 111	BH	DI

Register Specified by r/m During 32-bit Data Operations		
mod r/m	Function of w Field	
	(when w = 0)	(when w = 1)
11 000	AL	EAX
11 001	CL	ECX
11 010	DL	EDX
11 011	BL	EBX
11 100	AH	ESP
11 101	CH	EBP
11 110	DH	ESI
11 111	BH	EDI





## INSTRUCTION SET SUMMARY

Table E-9. Encoding of 32-bit Address Mode ("mod r/m" Byte and s-i-b Byte Present)

mod r/m	Effective Address
00 000	DS:[EAX + (scaled index)]
00 001	DS:[ECX + (scaled index)]
00 010	DS:[EDX + (scaled index)]
00 011	DS:[EBX + (scaled index)]
00 100	SS:[ESP + (scaled index)]
00 101	DS:[d32 + (scaled index)]
00 110	DS:[ESI + (scaled index)]
00 111	DS:[EDI + (scaled index)]
01 000	DS:[EAX + (scaled index) + d8]
01 001	DS:[ECX + (scaled index) + d8]
01 010	DS:[EDX + (scaled index) + d8]
01 011	DS:[EBX + (scaled index) + d8]
01 100	SS:[ESP + (scaled index) + d8]
01 101	SS:[EBP + (scaled index) + d8]
01 110	DS:[ESI + (scaled index) + d8]
01 111	DS:[EDI + (scaled index) + d8]
10 000	DS:[EAX + (scaled index) + d32]
10 001	DS:[ECX + (scaled index) + d32]
10 010	DS:[EDX + (scaled index) + d32]
10 011	DS:[EBX + (scaled index) + d32]
10 100	SS:[ESP + (scaled index) + d32]
10 101	SS:[EBP + (scaled index) + d32]
10 110	DS:[ESI + (scaled index) + d32]
10 111	DS:[EDI + (scaled index) + d32]

**NOTE:** Mod field in "mod r/m" byte; ss, index, base fields in "s-i-b" byte.

ss	Scale Factor
00	x1
01	x2
10	x4
11	x8

Index	Index Register
000	EAX
001	ECX
010	EDX
011	EBX
100	no index reg†
101	EBP
110	ESI
111	EDI

† When index field is 100, indicating "no index register," the ss field must equal 00. If this is not true, the effective address is undefined



### E.2.2.5 Encoding of Operation Direction (d) Field

In many two-operand instructions the d field is present to indicate which operand is considered the source and which is the destination.

**Table E-10. Encoding of Operation Direction (d) Field**

d	Direction of Operation
0	Register/Memory←Register "reg" field indicates source operand; "mod r/m" or "mod ss index base" indicates destination operand.
1	Register←Register/Memory "reg" field indicates destination operand; "mod r/m" or "mod ss index base" indicates source operand.

### E.2.2.6 Encoding of Sign-Extend (s) Field

The s field occurs primarily to instructions with immediate data fields. The s field has an effect only if the size of the immediate data is 8 bits and is being placed in a 16-bit or 32-bit destination.

**Table E-11. Encoding of Sign-Extend (s) Field**

s	Effect on Immediate Data8	Effect on Immediate Data 16 32
0	None	None
1	Sign-Extend Data8 to fill 16-bit or 32-bit destination	None

### E.2.2.7 Encoding of Conditional Test (ttn) Field

For the conditional instructions (conditional jumps and set on condition), ttn is encoded with n indicating to use the condition (n=0) or its negation (n=1), and ttt giving the condition to test.

**Table E-12. Encoding of Conditional Test (ttn) Field**

Mnemonic	Condition	ttn
O	Overflow	0000
NO	No Overflow	0001
B/NAE	Below/Not Above or Equal	0010
NB/AE	Not Below/Above or Equal	0011
E/Z	Equal/Zero	0100
NE/NZ	Not Equal/Not Zero	0101
BE/NA	Below or Equal/Not Above	0110
NBE/A	Not Below or Equal/Above	0111
S	Sign	1000
NS	Not Sign	1001
P/PE	Parity/Parity Even	1010
NP/PO	Not Parity/Parity Odd	1011
L/NGE	Less Than/Not Greater or Equal	1100
NL/GE	Not Less Than/Greater or Equal	1101
LE/NG	Less Than or Equal/Greater Than	1110
NLE/G	Not Less or Equal/Greater Than	1111



## INSTRUCTION SET SUMMARY

**E.2.2.8 Encoding of Control or Debug or Test Register (eee) Field**

For the loading and storing of the Control, Debug and Test registers.

**Table E-13. When Interpreted as Control Register Field**

eee Code	Reg Name
000	CR0
010	CR2
011	CR3

**NOTE:** Do not use any other encoding

**Table E-14. When Interpreted as Debug Register Field**

eee Code	Reg Name
000	DR0
001	DR1
010	DR2
011	DR3
110	DR6
111	DR7

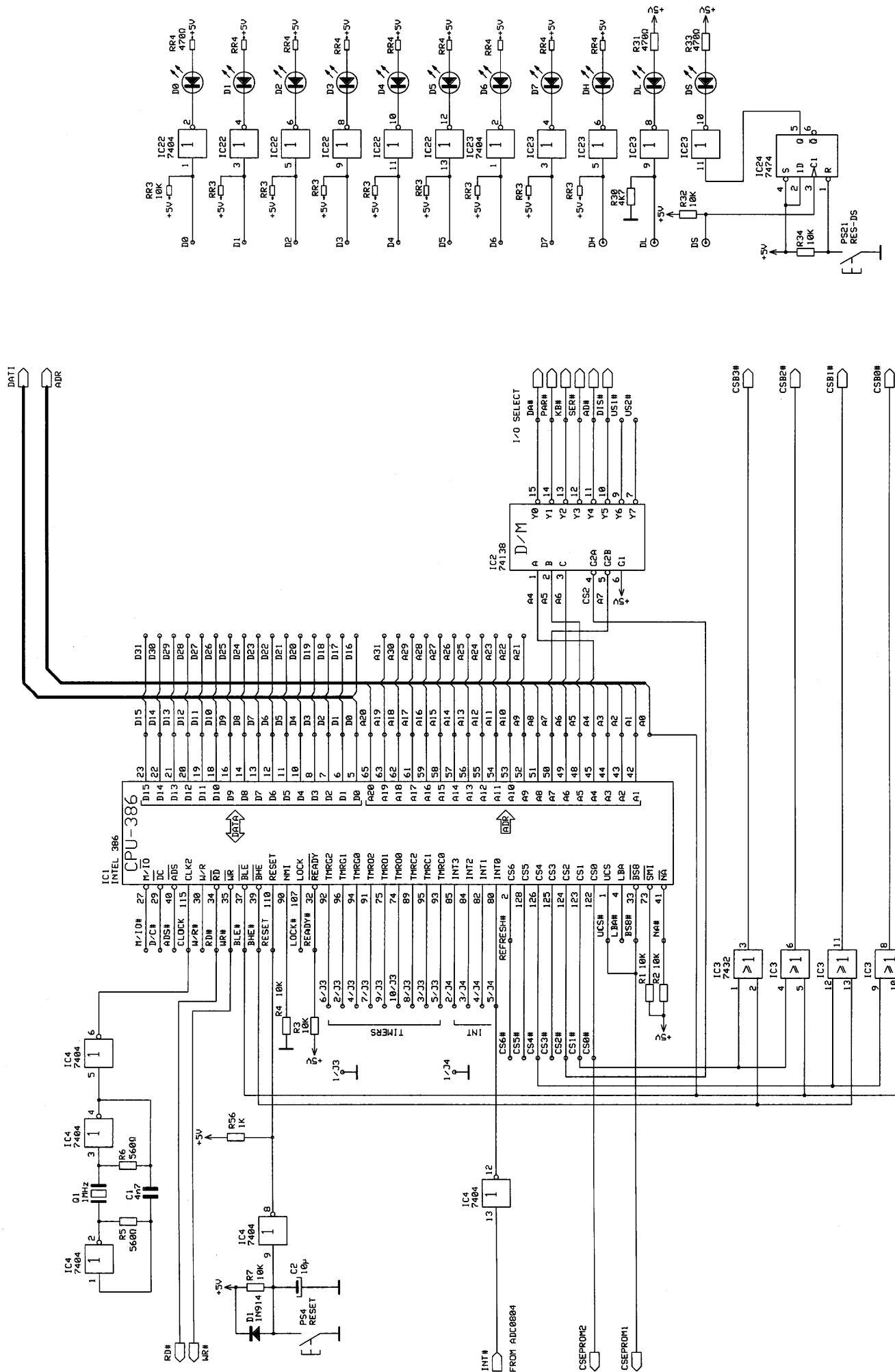
**NOTE:** Do not use any other encoding

**Table E-15. When Interpreted as Test Register Field**

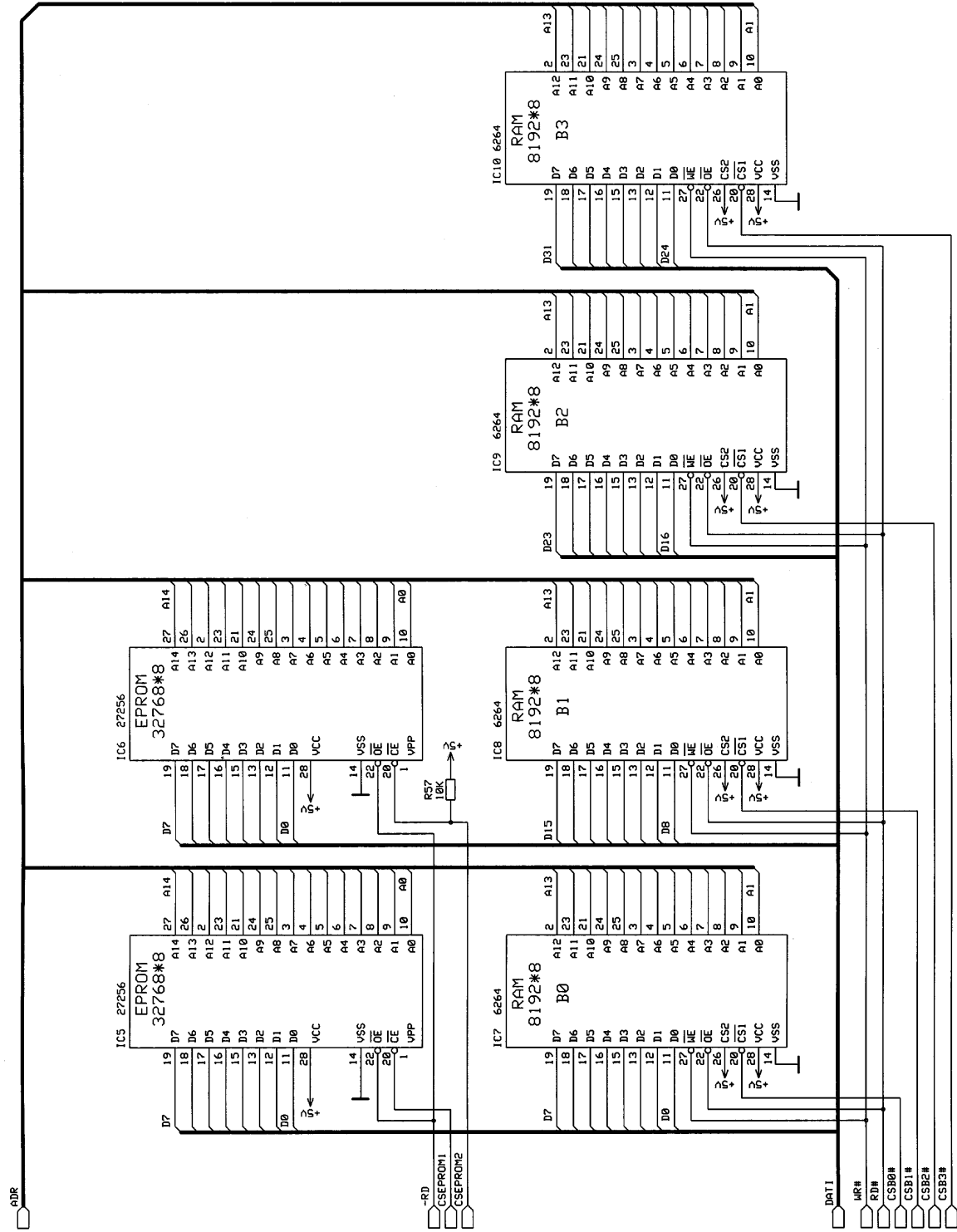
eee Code	Reg Name
110	TR6
111	TR7

**NOTE:** Do not use any other encoding

## **F. ELECTRICAL DIAGRAMS**



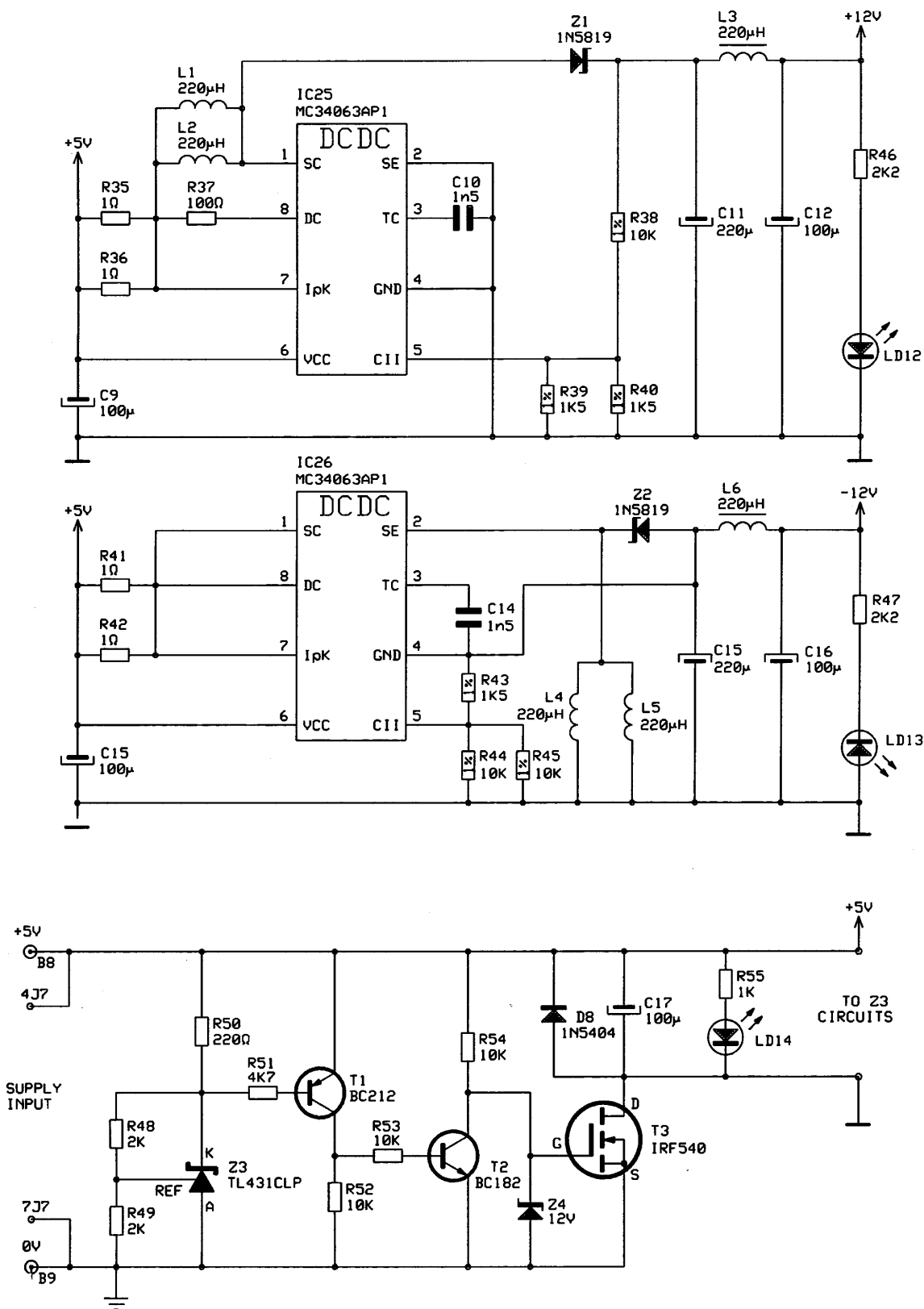
ELETTRONICA VENETA MOTTA DI LIVIGNO - TV ITALY	DESCRIPTION : 32 BIT MICROPROCESSOR		REPLACES : 235501	DMC N°: 23551001.FSH	DRAWN : <i>Barbieri</i>
	EQUIPMENT : MODULE 23/EV		REPLACED BY : --	P.C.B.: 235511	CHECKED : <i>Barbieri</i>
	T.C. FILE : --		REVISION: 2	SHEET : 1 OF 4	DATE : 22-11-00



ELETTRONICA VENETA MOTTA DI LIVENZA - TV ITALY	DESCRIPTION : MEMORY UNIT		REPLACES : Z3551D01.FSH	DRAWN : <i>[Signature]</i>
	EQUIPMENT : MODULE Z3/ev		REPLACED BY: ---	CHECKED : <i>[Signature]</i>
	T.C. FILE : ---		REVISION: 2	DATE : 22-11-00

SHEET : 2 OF 4	
----------------	--





ELETTRONICA VENETA MOTTA DI LIVIGNA - TV ITALY	DESCRIPTION : POWER SUPPLY		REPLACES : Z3\$01	DWG N°: Z3\$SID01.FSH	DRAWN : <i>Belus</i>
	EQUIPMENT : MODULE Z3/EV		REPLACED BY : --	P.C.B.: Z3\$11	CHECKED : <i>DP</i>
	T.C. FILE : --		REVISION: 2	SHEET : 4 OF 4	DATE : 22-11-00
	SCALE : --				