NABU-1200 SYSTEM
ENGINEERING
SPEC.
ENGINEERING SPECIFICATION
NABU 1200 SYSTEM

UNDATED COPY

NABU 1200 SYSTEM
TABLE OF CONTENT

1.0 SCOPE 4
2.0 APPLICABLE DOCUMENTS 4
3.0 GENERAL DESCRIPTION 5
4.0 BLOCK DIAGRAM 7
4.1 Description 7
5.0 CENTRAL PROCESSING UNIT 8
5.1 Description 8
5.2 Programming Information 8
6.0 NUMERIC DATA CO-PROCESSOR 8
6.1 Description 8
6.2 Programming Information 8
7.0 BUS ARBITER 9
7.1 Description 9
8.0 BUS CONTROLLER 9
8.1 Description 9
9.0 READ-ONLY MEMORY (ROM) 9
9.1 Description 9
10.0 READ-WRITE MEMORY (RAM) 10
10.1 Description 10
11.0 PROGRAMMABLE INTERVAL TIMER 10
11.1 Description 10
11.2 Programming Information 11
11.3 Examples 12
# TABLE OF CONTENT

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0</td>
<td>INTERRUPT CONTROLLER</td>
<td>13</td>
</tr>
<tr>
<td>12.1</td>
<td>Description</td>
<td>13</td>
</tr>
<tr>
<td>12.2</td>
<td>Programming Information</td>
<td>15</td>
</tr>
<tr>
<td>12.3</td>
<td>Examples</td>
<td>20</td>
</tr>
<tr>
<td>13.0</td>
<td>REAL TIME CLOCK</td>
<td>22</td>
</tr>
<tr>
<td>13.1</td>
<td>Description</td>
<td>22</td>
</tr>
<tr>
<td>13.2</td>
<td>Programming Information</td>
<td>22</td>
</tr>
<tr>
<td>13.3</td>
<td>Examples</td>
<td>22</td>
</tr>
<tr>
<td>14.0</td>
<td>INPUT/OUTPUT PORTS</td>
<td>23</td>
</tr>
<tr>
<td>14.1</td>
<td>Description</td>
<td>23</td>
</tr>
<tr>
<td>14.2</td>
<td>Programming Information</td>
<td>24</td>
</tr>
<tr>
<td>14.3</td>
<td>Examples</td>
<td>29</td>
</tr>
<tr>
<td>15.0</td>
<td>DIRECT MEMORY ACCESS (DMA) CONTROLLER</td>
<td>31</td>
</tr>
<tr>
<td>15.1</td>
<td>Description</td>
<td>31</td>
</tr>
<tr>
<td>15.2</td>
<td>Programming Information</td>
<td>32</td>
</tr>
<tr>
<td>15.3</td>
<td>Examples</td>
<td>36</td>
</tr>
<tr>
<td>16.0</td>
<td>FLOPPY DISK CONTROLLER</td>
<td>38</td>
</tr>
<tr>
<td>16.1</td>
<td>Description</td>
<td>38</td>
</tr>
<tr>
<td>16.2</td>
<td>Programming Information</td>
<td>38</td>
</tr>
<tr>
<td>16.3</td>
<td>Examples</td>
<td>39</td>
</tr>
<tr>
<td>17.0</td>
<td>HARD DISK CONTROLLER</td>
<td>41</td>
</tr>
<tr>
<td>17.1</td>
<td>Description</td>
<td>41</td>
</tr>
<tr>
<td>17.2</td>
<td>Programming Information</td>
<td>41</td>
</tr>
<tr>
<td>17.3</td>
<td>Examples</td>
<td>42</td>
</tr>
<tr>
<td>18.0</td>
<td>PERIPHERAL CONTROLLER</td>
<td>45</td>
</tr>
<tr>
<td>18.1</td>
<td>Description</td>
<td>45</td>
</tr>
<tr>
<td>18.2</td>
<td>Programming Information</td>
<td>46</td>
</tr>
<tr>
<td>18.3</td>
<td>Examples</td>
<td>47</td>
</tr>
<tr>
<td>19.0</td>
<td>MEMORY MANAGEMENT UNIT (MMU)</td>
<td>47</td>
</tr>
<tr>
<td>20.0</td>
<td>MASS STORAGE UNIT</td>
<td>47</td>
</tr>
</tbody>
</table>
1.0 SCOPE

This specification describes the technical aspect of the NABU 1200 system. This document is directed towards individuals with an appropriate technical background who wish to know more than the performance specifications of the system.

2.0 APPLICABLE DOCUMENTS

For additional information refer to the following publications:

NABU 1200 Technical Manual

NABU 1201 Memory Management Unit Engineering Specification

NABU 1210 Mass Storage Unit Specification

Hard Disk Controller Board Engineering Specification

Intel iAPX86 Product Description

Intel iAPX86 User's Manual

The 8086 Primer: Morse Hayden Book Co. 1980.
3.0 GENERAL DESCRIPTION

The NABU 1200 system is based on the Intel 8086 microprocessor. The CPU is a compact, single board 16 bit microprocessor. The system features random access memory, read-only memory, input/output ports, DMA transfer mechanism, interrupt controller, floppy disk drive controller, hard disk controller, mass storage unit, memory management unit and on-board self-test firmware routines.

A clock generator circuit provides a reliable reference to synchronize the various tasks of the system. Clock frequency is at 4.9152 MHz, resulting in fast execution of instructions.

The NABU 1200 is a portable, light-weight system with built-in expansion capabilities. The optional Memory Management Unit, designed for efficient use for operating systems, can be easily inserted into the basic CPU unit. Other features include user-selectable operating systems (CP/M, MS-DOS, XENIX) without any hardware changes.
4.9 BLOCK DIAGRAM

4.1 DESCRIPTION

Figure 1 is a block diagram of the NABU 1200 CPU board. The diagram consists of the following components:

- central processing unit
- numeric data co-processor (optional)
- bus arbiter
- bus controller
- read-only memory (ROM)
- read/write memory (RAM)
- programmable timers
- interrupt controller
- real time clock
- input/output ports
- direct memory access (DMA)
- floppy disk drive controller
- hard disk drive controller
- peripheral controller
- memory management unit (MMU)
- mass storage unit (MSU)

The following sections describe the above components in detail. Programming information is also provided with examples to aid understanding.
5.0 CENTRAL PROCESSING UNIT

5.1 DESCRIPTION

The processor board contains an Intel 8086 or equivalent microprocessor chip. The 8086 microprocessor features 16 bit word size with a maximum addressable memory capacity of one million bytes (1 Mbytes). The memory locations are word or byte addressable. Instruction length of the processor is variable in byte increment. The microprocessor runs at a clock rate of 4.9152 MHz. For more information refer to the applicable publications described.

5.2 PROGRAMMING INFORMATION

Refer to the applicable publications described above.

6.0 NUMERIC DATA CO-PROCESSOR

6.1 DESCRIPTION

Provision has been made on the printed circuit board to connect an 8087 floating point co-processor. This device provides high speed floating point arithmetic functions, effectively expanding the numeric processing capabilities of the system.

6.2 PROGRAMMING INFORMATION

Refer to the applicable publications described above.
7.0 BUS ARBITER

7.1 DESCRIPTION

The Bus Arbiter is an 8289 chip, which co-ordinates processors to operate asynchronously with one another. Each processor is assigned a different priority and the Bus Arbiter grants the bus to the processor with the highest priority.

8.0 BUS CONTROLLER

8.1 DESCRIPTION

The 8288 Bus Controller provides sophisticated bus control and command outputs. The Bus Controller, rather than the CPU, provides all control signals for the system bus and works together with the Bus Arbiter in a multibus configuration. This configuration allows CPU to support multiple processors on the system bus and provides compatibility with multibus architecture.

9.0 READ-ONLY MEMORY (ROM)

9.1 DESCRIPTION

The NABU 1200 system is supplied with two 2732 read-only memory chips pre-programmed with a bootstrap and self-test firmware. Each 2732 chip contains 4 Kbytes (8 bit byte) of memory, giving a total capacity of 8 Kbytes.

The chip will retrieve the content of a location specified by the address pins and place this information on the data pins. The memory locations are addressable from FE000 (hex) to FFFFFF (hex). Access time is 290 ns.
10.0 READ/WRITE MEMORY (RAM)

10.1 DESCRIPTION

Read/write memory is composed of 4864 chips, each of which provides 65K X 1 bit RAM. Current memory capacity is 256 Kbytes (32 4864 chips, 8 bit byte) of on-board RAM with parity checking; this is optionally expandable to 512 Kbytes. Access time is 150 ns.

11.0 PROGRAMMABLE INTERVAL TIMER

11.1 DESCRIPTION

There are two 8253 programmable interval timers. Each timer contains three independent 16-bit counters, giving a total of six counters for the system. The counters are used to control real time clock and input/output port baud rates.

The register assignments are as follows:

First timer:

<table>
<thead>
<tr>
<th>Register</th>
<th>Device controlled</th>
</tr>
</thead>
<tbody>
<tr>
<td>10H</td>
<td>real time clock</td>
</tr>
<tr>
<td>11H</td>
<td>diagnostic</td>
</tr>
<tr>
<td>12H</td>
<td>console ports</td>
</tr>
<tr>
<td>13H</td>
<td>control register</td>
</tr>
</tbody>
</table>

Second timer:

<table>
<thead>
<tr>
<th>Register</th>
<th>Device controlled</th>
</tr>
</thead>
<tbody>
<tr>
<td>16H</td>
<td>printer port</td>
</tr>
<tr>
<td>17H</td>
<td>communication port</td>
</tr>
<tr>
<td>18H</td>
<td>control register</td>
</tr>
</tbody>
</table>
11.2 Programming Information

Each counter of the 8253 is individually programmed by writing a control word into the Control Register. The format of the control word (8 bit byte) is:

\[
\begin{array}{cccccccc}
7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\
SC1 & SC2 & RL1 & RL2 & M2 & M1 & M0 & BCD \\
\end{array}
\]

Definition of control:

SC - select counter

<table>
<thead>
<tr>
<th>SC1</th>
<th>SC2</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>select counter 0 (registers 10H, 1CH)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>select counter 1 (registers 11H, 1DH)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>select counter 2 (registers 12H, 1EH)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>illegal</td>
</tr>
</tbody>
</table>

RL - Read/Load

<table>
<thead>
<tr>
<th>RL1</th>
<th>RL0</th>
<th>functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>read/load least significant byte first, then most significant byte</td>
</tr>
</tbody>
</table>

M - Mode

<table>
<thead>
<tr>
<th>M2</th>
<th>M1</th>
<th>M0</th>
<th>functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>square wave rate generator</td>
</tr>
</tbody>
</table>

BCD - counter value format

<table>
<thead>
<tr>
<th>BCD</th>
<th>functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>binary counter 16-bits</td>
</tr>
<tr>
<td>1</td>
<td>binary coded decimal (BCD) counter (4 decades)</td>
</tr>
</tbody>
</table>
11.3 Examples

To set the clock rate of the real time clock

```
mov al,00111110B ; select counter 0, binary counter
out 13H,al      ; send to first 8253 chip control reg.
                ; i.e. selecting the real time clock
mov ax,clockrate ; get binary value of clock rate
out 10H,al      ; send value to real time clock reg.
mov al,ah
out 10H,al
```

Note that once the control has been set up for a counter, it does not need to be set up again. The counter rate can be changed anytime afterward.
12.0 INTERRUPT CONTROLLER

12.1 DESCRIPTION

There are 256 interrupts available in the system. The interrupt number ranges from 0 to 255. The interrupt vectors are located from location 0 to 3FFH of the memory. Each vector occupies two words. The first word is the interrupt handler address offset, the second word is the segment. The first 240 interrupts are non-maskable, the rest are maskable by user.

A programmable interrupt controller enhances the interrupt capabilities of the 8086 microprocessor. The 8259A interrupt controller manages eight levels of requests and has built-in features for expandability to other 8259A's.

Two 8259A's are used in the system. One acts as a master and the other as a slave, together they provide 16 maskable interrupts. Their number and use are listed below in decreasing priority:

Master controller:

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>248</td>
<td>numeric data co-processor</td>
</tr>
<tr>
<td>249</td>
<td>memory parity error</td>
</tr>
<tr>
<td>250</td>
<td>system call</td>
</tr>
<tr>
<td>251</td>
<td>high-speed device operation</td>
</tr>
<tr>
<td>252</td>
<td>hard disk operation</td>
</tr>
<tr>
<td>253</td>
<td>floppy disk operation</td>
</tr>
<tr>
<td>254</td>
<td>real time clock</td>
</tr>
<tr>
<td>255</td>
<td>slave controller</td>
</tr>
</tbody>
</table>

Slave controller:

<table>
<thead>
<tr>
<th>Interrupt</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>240</td>
<td>serial interface port 0 input</td>
</tr>
<tr>
<td>241</td>
<td>serial interface port 0 output</td>
</tr>
<tr>
<td>242</td>
<td>serial interface port 1 input</td>
</tr>
<tr>
<td>243</td>
<td>serial interface port 1 output</td>
</tr>
<tr>
<td>244</td>
<td>serial interface port 2 input</td>
</tr>
<tr>
<td>245</td>
<td>serial interface port 2 output</td>
</tr>
<tr>
<td>246</td>
<td>serial interface port 3 input</td>
</tr>
<tr>
<td>247</td>
<td>serial interface port 3 output</td>
</tr>
</tbody>
</table>
Command registers

There are two types of command, initialization and operation. The corresponding registers are:

Master controller:

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>40H</td>
<td>initialization command word 1</td>
</tr>
<tr>
<td></td>
<td>operation command word 2</td>
</tr>
<tr>
<td>42H</td>
<td>initialization command word 2</td>
</tr>
<tr>
<td></td>
<td>initialization command word 3</td>
</tr>
<tr>
<td></td>
<td>initialization command word 4</td>
</tr>
<tr>
<td></td>
<td>operation command word 1</td>
</tr>
</tbody>
</table>

Slave controller:

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>50H</td>
<td>initialization command word 1</td>
</tr>
<tr>
<td></td>
<td>operation command word 2</td>
</tr>
<tr>
<td>52H</td>
<td>initialization command word 2</td>
</tr>
<tr>
<td></td>
<td>initialization command word 3</td>
</tr>
<tr>
<td></td>
<td>initialization command word 4</td>
</tr>
<tr>
<td></td>
<td>operation command word 1</td>
</tr>
</tbody>
</table>
12.2 PROGRAMMING INFORMATION

Command Word Formats

Initialization Command Words

Word 1

<table>
<thead>
<tr>
<th>bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>LTM</td>
<td>ADI</td>
<td>SNG</td>
</tr>
</tbody>
</table>

Definition of control:

LTM - trigger mode

1 = level triggered mode
0 = edge triggered mode (use this value)

ADI - call address interval (not used under 8086 system)

1 = interval of 4 bytes
0 = interval of 8 bytes (use this value)

SNG - single

1 = single mode
0 = cascade mode (use this value)

IC4 - initialization word 4

1 = initialization word 4 needed (use this value)
0 = initialization word 4 not needed

Word 2

<table>
<thead>
<tr>
<th>bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>T7</td>
<td>T6</td>
<td>T5</td>
<td>T4</td>
<td>T3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Definition of control:

T7 - T3 = first interrupt value of controller

= 11111 for master controller (interrupt 248)
= 11110 for slave controller (interrupt 240)
Word 3

Master controller:

<table>
<thead>
<tr>
<th>bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S7</td>
<td>S6</td>
<td>S5</td>
<td>S4</td>
<td>S3</td>
<td>S2</td>
<td>S1</td>
<td>S0</td>
</tr>
</tbody>
</table>

Definition of control:

S - controller has slave at this interrupt

1 = has slave to controller
0 = no slave to controller

use S7 = 1, S6 - S0 = 0

Slave controller:

<table>
<thead>
<tr>
<th>bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>ID2</td>
<td>ID1</td>
</tr>
</tbody>
</table>

Definition of control:

ID2 - ID0 = slave ID number, corresponding to location of cascading slave on master controller

= 111 for this system (use this value)
Word 4

bits 7 6 5 4 3 2 1 0

0 0 0 SNM BUF M/S EOI uPM

Definition of control:

SNM - special fully nested

1 = special fully nested mode
0 = not special fully nested mode (use this value)

BUF, M/S - buffered master/slave

0X = non-buffered mode (use this value)
10 = buffered mode/slave
11 = buffered mode/master

EOI - end of interrupt

1 = automatic end of interrupt
0 = normal end of interrupt (use this value)

uPM - microprocessor mode

1 = 8086/8088 mode (use this value)
0 = MCS-86/85 mode
Operation Command Words

After the initialization of the interrupt controller, operation commands can be sent to the controller anytime to set up the operation modes.

Word 1

bits    7   6   5   4   3   2   1   0
M7  M6  M5  M4  M3  M2  M1  M0

Definition of control:

M7 - M0 = interrupt mask, M7 for high interrupt number, M0 for low interrupt number, one bit corresponds to one interrupt location
1 = mask set
0 = mask reset

Word 2

bits    7   6   5   4   3   2   1   0
R  SL  EOI  0  0  L2  L1  L0

Definition of control:

R, SL, EOI - rotate, specific, end of interrupt

001 = non-specific EOI command
011 = specific EOI command
101 = rotate on non-specific EOI command
100 = rotate in automatic EOI mode (set)
000 = rotate in automatic EOI mode (clear)
111 = rotate on specific EOI command
110 = set priority command
010 = no operation

L2 - L0 = specific interrupt level in controller to be acted on when SL is set
Word 3

bits 7 6 5 4 3 2 1 0

0 ESM SMM 0 1 P RR RIS

Definition of control:

ESM, SMM - enable special mask mode, special mask mode

00 = no action
01 = no action
10 = reset special mask
11 = set special mask

P - poll command

1 = poll command
0 = no poll command

RR, RIS - read register command

00 = no action
01 = no action
10 = read IR register on next RD pulse
11 = read IS register on next RD pulse
12.3 Examples

To initialize the master controller

```plaintext
mov al, 00010001B ; initialization control word 1
out 40H, al
mov al, 11111000B ; initialization control word 2
out 42H, al
mov al, 10000000B ; initialization control word 3
out 42H, al
mov al, 00000001B ; initialization control word 4
out 42H, al
```

To initialize the slave controller

```plaintext
mov al, 00010001B ; initialization control word 1
out 50H, al
mov al, 11110000B ; initialization control word 2
out 52H, al
mov al, 00000111B ; initialization control word 3
out 52H, al
mov al, 00000001B ; initialization control word 4
out 52H, al
```

To enable interrupt 240 to 247

```plaintext
mov al, intmask ; operation control word 1
out 52H, al
```

To enable interrupt 248 to 255

```plaintext
mov al, intmask ; operation control word 1
out 42H, al
```

Note that in order for any of the interrupts on the slave interrupt controller to be enabled, the interrupt 255 on the master controller must be also be enabled.
Interrupt handling

Hard disk interrupt handling

; immediate before interrupt return

in al, 77H ; get status and clear interrupt
mov al, 01100100B ; on controller board
out 40H, al ; operation command word 2
           ; (end of interrupt command)

Floppy disk interrupt handling

; immediate before interrupt return

in al, 1BH ; get status and clear interrupt
mov al, 01100101B ; on controller board
out 40H, al ; operation command word 2
           ; (end of interrupt command)

Real time clock interrupt handling

; immediate before interrupt return

mov al, 01100110B ; operation command word 2
out 40H, al ; (end of interrupt command)

Serial interface interrupt handling

using port 0 as example:

; immediate before interrupt return

mov al, 01100111B ; operation command word 2
out 40H, al ; (end of interrupt command)
mov al, 01100000B ; clear master interrupt
out 50H, al ; (end of interrupt command)
mov al, 00H ; clear slave interrupt
in al, 00H
and al, 01111111B ; get data from port 0 data reg.
-Parity-error-interrupt-handling-

-out---D0H,al--- ; clear interrupt
13.0 REAL TIME CLOCK

13.1 DESCRIPTION

The real time clock is set up using one of the counter of the first timer (see section 11). The interrupt clock rate can be changed anytime after setting up of the counter control.

13.2 PROGRAMMING INFORMATION

The following equation is used to determine the real time clock parameter value:

\[ \frac{4915200 \text{ / clock rate in cycles}}{2} \]

then convert the number to a hexadecimal number.

13.3 EXAMPLES

To set the real time clock to interrupt 60 times per second

The parameter value is:

\[ \frac{4915200 \text{ / } 60 \text{ / } 2}{2} = 40960 \]

When converted to a hexadecimal number, it is A000H.

\begin{verbatim}
  mov  a1,001111110B ; select counter 0, mode 3, binary
  out  13H,a1      ; send to first timer control register
  mov  a1,00H      ; get lower byte of parameter
  out  10H,a1      ; send to counter 0 register
  mov  a1,A0H      ; get upper byte of parameter
  out  10H,a1      ; send to counter 0 register
\end{verbatim}
14.0 INPUT/OUTPUT PORT

14.1 DESCRIPTION

Four 8251A programmable communication interface chips are used to provide four RS-232C asynchronous serial input/output ports. Two of the ports can be configured for synchronous operation.

The 8253 programmable timers described in section 11 are used to set the baudrate of the ports anywhere between 50 and 19200 baud.

The addresses of the ports are:

<table>
<thead>
<tr>
<th>Port</th>
<th>Status/Command Reg.</th>
<th>Data Reg.</th>
<th>Timer Reg.</th>
<th>Counter</th>
<th>Interrupt Input</th>
<th>Interrupt Output</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>01H</td>
<td>00H</td>
<td>12H</td>
<td>2</td>
<td>240</td>
<td>241</td>
<td>Printer</td>
</tr>
<tr>
<td>1</td>
<td>05H</td>
<td>04H</td>
<td>1CH 11H</td>
<td>0</td>
<td>242</td>
<td>243</td>
<td>Console/Printer</td>
</tr>
<tr>
<td>2</td>
<td>09H</td>
<td>08H</td>
<td>1DH 11H</td>
<td>1</td>
<td>244</td>
<td>245</td>
<td>Console/Communication</td>
</tr>
<tr>
<td>3</td>
<td>0DH</td>
<td>0CH</td>
<td>1EH 11H</td>
<td>2</td>
<td>246</td>
<td>247</td>
<td>Console/Communication</td>
</tr>
</tbody>
</table>
14.2 PROGRAMMING INFORMATION

Mode Instruction

Asynchronous Mode Instruction Format

bits  7  6  5  4  3  2  1  0
     S2  S1  EP  PEN  L2  L1  B2  B1

Definition of control:

S2, S1 - number of stop bits

00 = invalid
01 = 1 bit
10 = 1 1/2 bits
11 = 2 bits

EP - even parity

1 = even
0 = odd

PEN - parity enable

1 = enable
0 = disable

L2, L1 - character length

00 = 5 bits
01 = 6 bits
10 = 7 bits
11 = 8 bits

B2, B1 - baud rate factor

00 = synchronous mode
01 = 1X
10 = 16X
11 = 64X
Synchronous Mode Instruction Format

bits  7  6  5  4  3  2  1  0
     SCS  ESD  EP  PEN  L2  L1  0  0

Definition of control:

SCS - single character sync
    1 = single sync character
    0 = double sync character

ESD - external sync detect
    1 = syndet is an input
    0 = syndet is an output

EP - even parity generation/check
    1 = even
    0 = odd

PEN - parity enable
    1 = enable
    0 = disable

L2, L1 - character length
    00 = 5 bits
    01 = 6 bits
    10 = 7 bits
    11 = 8 bits
Command Instruction

bits 7 6 5 4 3 2 1 0

EH  IR RTS  ER BRK RXE DTR TXE

Definition of control:

EH - enter hunt mode
  1 = enable serach for Sync character

IR - internal reset
  1 = return to mode instruction format

RTS - request to send
  1 = force RTS output to zero

ER - error reset
  1 = reset error flags PE, OE, FE

BRK - send break
  1 = force TXD to low
  0 = normal operation

RXE - receive enable
  1 = enable
  0 = disable

DTR - data terminal ready
  1 = force DTR output to zero

TXE - transmit enable
  1 = enable
  0 = disable
Status Word Format

bits 7 6 5 4 3 2 1 0

DSR DET FE OE PE TXE RX TX

Definition of status:

DSR - data set ready

1 = DSR at a zero level

DET -

FE - framing error

1 = framing error detected

OE - overrun error

1 = data overrun

PE - parity error

1 = parity error detected

TXE - transmit buffer empty

1 = transmit buffer empty

RX - receiver ready

1 = data is ready to be input to the CPU

TX - transmit ready

1 = ready to accept data from the CPU
The following table should be used to determine the parameter for setting the baudrate for a port.

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>Parameter (hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>19200</td>
<td>0008</td>
</tr>
<tr>
<td>9600</td>
<td>0016</td>
</tr>
<tr>
<td>4800</td>
<td>0032</td>
</tr>
<tr>
<td>2400</td>
<td>0064</td>
</tr>
<tr>
<td>1200</td>
<td>0128</td>
</tr>
<tr>
<td>600</td>
<td>0256</td>
</tr>
<tr>
<td>300</td>
<td>0512</td>
</tr>
</tbody>
</table>

The parameter value is calculated using the equation:

\[
\frac{4915200}{\text{baudrate}} \div 16 \div 2
\]

and the result converted to a Binary Coded Decimal (BCD) number.
14.3 EXAMPLES

To calculate the baudrate parameter for 9600 baud

\[
4915200 \div 9600 \div 16 \div 2 = 16
\]

When converted to BCD, the value is 0016 (hex).

To set the baudrate for port 0 to 9600 baud

\[
\text{mov al,01111111B} \quad \text{mov al,16H} \quad \text{mov al,00H} \quad \text{mov al,00H} \\
\text{out 13H,al} \quad \text{out 12H,al} \quad \text{out 12H,al} \quad \text{out 12H,al}
\]

; select counter 1, BCD parameter value
; send to first timer control register
; get lower byte of parameter
; send to counter register
; get upper byte of parameter
; send to counter register

To initialize the input/output ports

Using port 0 for example:

\[
\text{mov al,10001110B} \quad \text{mov al,0110111B} \quad \text{mov al,01001110B} \\
\text{out 01H,al} \quad \text{out 01H,al} \quad \text{out 01H,al} \\
\text{nop} \quad \text{nop} \quad \text{nop} \\
\text{nop} \quad \text{nop} \quad \text{nop}
\]

; force interface into command mode
; send command for internal reset
; send mode instruction for async
; send to command register
; delay
; send to command register
; mode, 1 stop bit, disable parity,
; 8 bit data, 16X clock rate
; send to command register

\[
\text{mov al,00110111B} \quad \text{mov al,00110111B} \\
\text{out 01H,al} \quad \text{out 01H,al}
\]

; send command instruction to enable
; receiver and transmitter
The following are I/O programming examples for port 0, using polling method.

To check status for input

loopl:  in  al,01H ; get status from port 0
    and  al,00000010B ; check data available bit in status
    jz  loopl ; not set, loop and wait

To check status for output

loopl:  in  al,01H ; get status from port 0
    and  al,00000001B ; check if previous data sent
    jz  loopl ; not sent, loop and wait

To input from port

; check status for input first

    in  al,00H ; get data from port 0
    and  al,01111111H ; strip off parity bit

To output from port

; check status for output first

    out  00H,al ; send data to port 0
15.0 DIRECT MEMORY ACCESS (DMA) CONTROLLER

15.1 DESCRIPTION

The Programmable Direct Memory Access (DMA) controller 8237A provides four channels. These channels support high speed data transfer between high speed I/O devices and memory; without intervention by the CPU. Before the DMA transfer between the device and memory, the DMA channel must be set up first.

The channel assignments are as follow:

<table>
<thead>
<tr>
<th>Channel</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Floppy disk drive controller</td>
</tr>
<tr>
<td>1</td>
<td>Hard disk drive controller</td>
</tr>
<tr>
<td>2</td>
<td>Dynamic RAM refresh</td>
</tr>
<tr>
<td>3</td>
<td>reserved</td>
</tr>
</tbody>
</table>

The register assignments are as follow:

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>60H</td>
<td>floppy disk current address register</td>
</tr>
<tr>
<td>61H</td>
<td>floppy disk current transfer count register</td>
</tr>
<tr>
<td>62H</td>
<td>hard disk current address register</td>
</tr>
<tr>
<td>63H</td>
<td>hard disk current transfer count register</td>
</tr>
<tr>
<td>64H</td>
<td>RAM refresh current address register</td>
</tr>
<tr>
<td>65H</td>
<td>RAM refresh current count register</td>
</tr>
<tr>
<td>66H-</td>
<td>high-speed-device-current-address-register</td>
</tr>
<tr>
<td>67H-</td>
<td>high-speed-device-current-transfer-count-reg</td>
</tr>
<tr>
<td>68H</td>
<td>command/status register</td>
</tr>
<tr>
<td>69H</td>
<td>request register (write only)</td>
</tr>
<tr>
<td>6AH</td>
<td>mask set/rest register (write only)</td>
</tr>
<tr>
<td>6BH</td>
<td>mode register (write only)</td>
</tr>
<tr>
<td>6DH</td>
<td>temporary register</td>
</tr>
<tr>
<td>6FH</td>
<td>write all mask register</td>
</tr>
</tbody>
</table>
15.2 PROGRAMMING INFORMATION

Command Formats

Command Register

bits  7 6 5 4 3 2 1 0
     A B C D E F G H

Definition of control:

A = 0  DACK sense active low (use this value)
     = 1  DACK sense active high

B = 0  DREQ sense active high (use this value)
     = 1  DREQ sense active low

C = 0  late write selection
     = 1  extended write selection (use this value)
     = X  if bit 3 = 1

D = 0  fixed priority (use this value)
     = 1  rotating priority

E = 0  normal timing (use this value)
     = 1  compressed timing
     = X  if bit 0 = 1

F = 0  controller enable (use this value)
     = 1  controller disable

G = 0  channel 0 address hold disable (use this value)
     = 1  channel 0 address hold enable
     = X  if bit 0 = 0

H = 0  memory-to-memory disable (use this value)
     = 1  memory-to-memory enable
Mode Register

bits  7 6 5 4 3 2 1 0
      A B C D E F G H

Definition of control:

A, B = 00 demand mode select
       = 01 single mode select (use this value)
       = 10 block mode select
       = 11 cascade mode select

C = 0 address increment select (use this value)
       = 1 address decrement select

D = 0 autoinitialization disable (use this value)
       = 1 autoinitialization enable

E, F = 00 verify transfer
       = 01 write transfer
       = 10 read transfer
       = 11 illegal
       = XX if bits 6 and 7 = 11

G, H = 00 channel 0 select
       = 01 channel 1 select
       = 10 channel 2 select
       = 11 channel 3 select
Mask Register

Set/clear format:

bits  7  6  5  4  3  2  1  0
      X  X  X  X  X  A  B  C

Definition of control:

X = don't care

A = 0  clear mask bit
    = 1  set mask bit

B, C = 00  select channel 0 mask bit
      = 01  select channel 1 mask bit
      = 10  select channel 2 mask bit
      = 11  select channel 3 mask bit

Single command format:

bits  7  6  5  4  3  2  1  0
      X  X  X  X  A  B  C  D

X = don't care

A = 0  clear channel 0 mask bit
    = 1  set channel 0 mask bit

B = 0  clear channel 1 mask bit
    = 1  set channel 1 mask bit

C = 0  clear channel 2 mask bit
    = 1  set channel 2 mask bit

D = 0  clear channel 3 mask bit
    = 1  set channel 3 mask bit
MMU address byte

The addressing byte is used to specify the address area under MMU for DMA transfer.

\[
\begin{align*}
\text{bits} & \quad 7 \quad 6 \quad 5 \quad 4 \quad 3 \quad 2 \quad 1 \quad 0 \\
        & \quad 0 \quad 0 \quad A \quad B \quad C \quad D \quad E \quad F
\end{align*}
\]

- A = 0 \quad \text{lower 64K bytes}
- A = 1 \quad \text{upper 64K bytes}
- B = 0 \quad \text{code area}
- B = 1 \quad \text{data area}
- C, D, E, F = \text{task number (0000 to 1111)}

To set the appropriate address area, the following registers are used:

<table>
<thead>
<tr>
<th>Register</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>2AH</td>
<td>Floppy disk DMA transfer</td>
</tr>
<tr>
<td>2CH</td>
<td>Hard disk DMA transfer</td>
</tr>
</tbody>
</table>
15.3 Examples

To initialize the DMA controller

```asm
mov al,00100000B ; get command word
out 68H,al ; send to command register
```

Floppy disk DMA transfer set up

```asm
mov ax,bytecount ; get byte count (actual count - 1)
out 61H,al ; send out lower byte
mov al,ah ; get upper byte
out 61H,al ; send out higher byte

; calculate 20 bits buffer address,
; using offset and segment

mov dx,bufoffset ; get buffer address offset
mov ax,bufsegment ; get buffer address segment
mov cl,4 ; get shift count
shr dx,cl ; shift right 1 nibble
add ax,dx ; add to segment
mov dx,ax ; save first 16 bits
shl dx,cl ; get last 3 nibbles
mov cx,bufoffset ; get buffer address offset
and cx,000FH ; get offset last nibble
or dx,cx ; combine with the 3 nibbles ; to form the last 4 nibbles
mov cl,12 ; get shift count
shr ax,cl ; move first nibble to the right
out 14H,al ; send it out to controller
mov al,10H ; set address mode under MMU
out 2AH,al
mov ax,dx ; get last 4 nibbles
out 60H,al ; send lower byte
mov al,ah ; get upper byte
out 60H,al ; send upper byte
mov al,01000100B ; get floppy read mode
out 6BH,al ; use 01001000B for write mode
mov al,00000000B ; send to mode register
out 6AH,al ; clear channel mask
```
Hard disk DMA transfer set up

mov ax, bytecount ; get byte count (actual count -1)
out 63H, al ; send out lower byte
mov al, ah ; get upper byte
out 63H, al ; send out higher byte

; calculate 20 bit buffer address,
; using offset and segment

mov dx, bufoffset ; get buffer address offset
mov ax, bufsegment ; get buffer address segment
mov cl, 4 ; get shift count
shr dx, cl ; shift right 1 nibble
add ax, dx ; add to segment
mov dx, ax ; save first 16 bits
shl dx, cl ; get last 3 nibbles
mov cx, bufoffset ; get buffer address offset
and cx, 000FH ; get offset last nibble
or dx, cx ; combine with the 3 nibbles
; to form the last 4 nibbles
mov cl, 0-12 ; get shift count
shr ax, cl ; move first nibble to the right
out 14H, al ; send it out to controller
mov al, 10H ; set address mode under MMU
out 2CH, al
mov ax, dx ; get last 4 nibbles
out 62H, al ; send lower byte
mov al, ah ; get upper byte
out 62H, al ; send upper byte
mov al, 01000101B ; get hard disk read mode
out 6BH, al ; use 01000101B for write mode
mov al, 00000001B ; send to mode register
out 6AH, al ; clear channel mask
16.0 FLOPPY DISK CONTROLLER

16.1 DESCRIPTION

The floppy disk controller supports 4 floppy disk drives, drives 0 to 2 for mini-floppy, drive 3 for maxi-floppy.

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>16H</td>
<td>disk selection</td>
</tr>
<tr>
<td>18H</td>
<td>data</td>
</tr>
<tr>
<td>19H</td>
<td>sector</td>
</tr>
<tr>
<td>1AH</td>
<td>track</td>
</tr>
<tr>
<td>1BH</td>
<td>command</td>
</tr>
<tr>
<td>1BH</td>
<td>status</td>
</tr>
</tbody>
</table>

Drive select byte

\[
\begin{array}{cccccccc}
A & B & C & D & E & F & L & L \\
\hline
A = 1 & drive 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
B = 1 & drive 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
C = 1 & drive 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
D = 1 & drive 3 & 0 & 0 & 0 & 0 & 0 & 0 \\
E = 0 & side 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
F = 0 & side 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
F = 1 & 5 1/4" floppy, double density & 0 & 0 & 0 & 0 & 0 & 0 \\
F = 1 & 8" floppy, single density & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

16.2 PROGRAMMING INFORMATION

Refer to the Western Digital FD 179X-02 Floppy Disk Formatter/Controller Family Specification for details.
16.3 EXAMPLES

To select the floppy drive before use

\begin{verbatim}
mov al,10000011B ; drive 0
 ; side 0
 ; 512 bytes/sector
out 16H,al ; send select byte to controller
\end{verbatim}

To home (restore) floppy disk

\begin{verbatim}
mov al,00001100B ; get restore command
 ; use 00001100B for 8" drive only
out 1BH,al ; send to command register

; wait for interrupt return
in al,1BH ; get status
and al,10010000B ; check for error
jz continue ; no error, continue

; on error, retry
\end{verbatim}

To seek a track on floppy disk

\begin{verbatim}
mov al,lasttrack ; get last track access on drive
out 1AH,al ; send to track register
mov al,tracknum ; get track number to seek
out 18H,al ; send to data register
mov al,00010100B ; get seek command
 ; use 00010100B for 8" drive only
out 1BH,al ; send to command register

; wait for interrupt return
in al,1BH ; get status
and al,10010000B ; check for error
jz continue ; no error, continue

; on error, retry
\end{verbatim}
To read a sector from floppy disk

; set up DMA controller first (see section 15.0)

mov al,sectornum ; get sector number
out 19H,al ; send to sector register
mov al,10000100B ; get read sector command on side 0
; use 10001110B on side 1
out 1BH,al ; send to command register

; wait for interrupt return using polling

in al,1BH ; get status
and al,10011110B ; check for error
jz continue ; no error, continue

; on error, retry

To write a sector to floppy disk

; set up DMA controller first (see section 15.0)

mov al,sectornum ; get sector number
out 19H,al ; send to sector register
mov al,10100100B ; get write sector command on side 0
; use 10101110B on side 1
out 1BH,al ; send to command register

; wait for interrupt return using polling

in al,1BH ; get status
and al,11111110B ; check for error
jz continue ; no error, continue

; on error, retry

Note:

The following steps must be done to ensure successful floppy disk read/write operations:

1. A minimum of 21 msec. delay must be performed following a track seek operation.

2. A minimum of 2 msec. delay must be performed following a read/write operation.

3. A minimum of 200 usec. delay must be performed following a change of side select.
17.0 HARD DISK CONTROLLER

17.1 DESCRIPTION

The hard disk controller supports up to 4 hard disk drives.

<table>
<thead>
<tr>
<th>Register</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>70H</td>
<td>data</td>
</tr>
<tr>
<td>71H</td>
<td>error</td>
</tr>
<tr>
<td>71H</td>
<td>write precompensation</td>
</tr>
<tr>
<td>72H</td>
<td>sector count</td>
</tr>
<tr>
<td>73H</td>
<td>sector number</td>
</tr>
<tr>
<td>74H</td>
<td>cylinder low byte</td>
</tr>
<tr>
<td>75H</td>
<td>cylinder high byte</td>
</tr>
<tr>
<td>76H</td>
<td>sector size, drive, head</td>
</tr>
<tr>
<td>77H</td>
<td>status</td>
</tr>
<tr>
<td>77H</td>
<td>command</td>
</tr>
</tbody>
</table>

17.2 PROGRAMMING INFORMATION

Refer to the Hard Disk Controller documentation for details.
17.3 EXAMPLES

To home (restore) hard disk

```
mov al,10H ; get precompensation factor
out 71H,al ; send to write precompensation register
mov al,00100000B ; get select byte
    ; use 00101000B for drive 1
    ; use 00110000B for drive 2
    ; use 00111000B for drive 3
out 76H,al ; send to select register
mov al,00010110B ; get restore command
out 77H,al ; send to command register

; wait for interrupt return
in al,77H ; get status
and al,00000001B ; check for error
jz continue ; no error, continue

; on error, retry
```

To seek a track on hard disk

No seek is required for hard disk, implied seek is performed in read/write operation.
To read a sector from hard disk

; set up DMA controller first (see section 15.0)

mov al,sectornum   ; get sector number
out 73H,al         ; send to sector register
mov al,00100000B   ; get select byte for drive 0
mov dl,headnum     ; get head number
or al,dl           ; put head number into select byte
out 76H,al         ; send to select register
mov ax,cylindernum ; get cylinder number
out 74H,al         ; send cylinder number low byte
mov al,ah           ; get cylinder number high byte
out 75H,al         ; send cylinder number high byte
mov al,000101000B   ; get read sector command
out 77H,al         ; send to command register

; wait for interrupt return using polling

in  al,77H          ; get status
and al,00000001H    ; check for error
jz    continue      ; no error, continue

; on error, retry

To write a sector to hard disk

; set up DMA controller first (see section 15.0)

mov al,sectornum   ; get sector number
out 73H,al         ; send to sector register
mov al,00100000B   ; get select byte for drive 0
mov dl,headnum     ; get head number
or al,dl           ; put head number into select byte
out 76H,al         ; send to select register
mov ax,cylindernum ; get cylinder number
out 74H,al         ; send cylinder number low byte
mov al,ah           ; get cylinder number high byte
out 75H,al         ; send cylinder number high byte
mov al,001110000B   ; get write sector command
out 77H,al         ; send to command register

; wait for interrupt return using polling

in  al,77H          ; get status
and al,00000001H    ; check for error
jz    continue      ; no error, continue

; on error, retry
To wait for floppy disk interrupt return using polling method

loop1:
  in  al,1BH          ; get status
  test al,00000001B   ; check if busy
  jz   loop1          ; no, loop

loop2:
  in  al,1BH          ; get status
  test al,00000001B   ; check if busy
  jnz  loop2          ; yes, loop

To wait for hard disk interrupt return under polling method

loop1:
  in  al,77H          ; get status
  test al,10000000B   ; check if busy
  jz   loop1          ; no, loop

loop2:
  in  al,77H          ; get status
  test al,10000000B   ; check if busy
  jnz  loop2          ; yes, loop

Error messages for hard disk

When error indicator is returned in the status word, use the following to get the error word:

  in  al,71H          ; get the error word
### 18.0 PERIPHERAL CONTROLLER

#### 18.1 DESCRIPTION

The 8255 peripheral controller is used for special purposes. There are three ports available on the controller. They are:

<table>
<thead>
<tr>
<th>Register</th>
<th>I/O</th>
<th>Port</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>14H</td>
<td>output</td>
<td>A</td>
<td>most significant 4 bits of 20-bit DMA transfer address for hard and floppy disk (bits 7 – 4 for hard disk, bits 3 – 0 for floppy disk)</td>
</tr>
<tr>
<td>15H</td>
<td>input</td>
<td>B</td>
<td>Machine serial number</td>
</tr>
<tr>
<td>16H</td>
<td>output</td>
<td>C</td>
<td>Floppy disk drive selection (see section 16)</td>
</tr>
<tr>
<td>17H</td>
<td></td>
<td></td>
<td>Mode selection</td>
</tr>
</tbody>
</table>
18.2 PROGRAMMING INFORMATION

Mode Definition Format

bits  7   6   5   4   3   2   1   0
     A   B   C   D   E   F   G   H

A - mode set flag
    1 = active

B, C - mode selection
    00 = mode 0, basic input/output (use this value)
    01 = mode 1, strobed input/output
    1x = mode 2, bi-directional bus

D - port A
    1 = input
    0 = output (use this value)

E - port C, upper
    \  
    1 = input
    0 = output (use this value)

F - mode selection
    0 = mode 0 (use this value)
    1 = mode 1

G - port B
    1 = input (use this value)
    0 = output

H - port C, lower
    1 = input
    0 = output (use this value)
18.3 EXAMPLES

To initialize the peripheral controller

```
mov al,10000010B ; set port A to output
                ; port B to input
                ; port C to output
out 17H,al      ; send to controller
```

19.0 MEMORY MANAGEMENT UNIT (MMU)

Refer to the NABU 1201 Memory Management Unit Engineering Specification for more details.

20.0 MASS STORAGE UNIT (MSU)

Refer to the NABU 1210 Mass Storage Unit Specification for more details.