NABU-1200 MMU

ENGINEERING

SPEC.

TABLE OF CONTENTS

TABLES

1.0 SCOPE

This specification describes the operation of the optional Memory Management Unit (MMU) used with the NABU-1200 system.

2.0 APPLICABLE DOCUMENTS

For additional information, refer to the following publications:

NABU-1200 Engineering Specification.

3.0 GENERAL DESCRIPTION

The MMU consists of a printed circuit board which plugs into the slot provided on the NABU-1200 CPU board.

The MMU performs the following functions:

1. Provides the operating system with multitasking capability and memory access protection.
2. Dynamically allocates a maximum of 256 Kbytes of memory to each user in a multi-user environment (128 Kbytes for data + 128 Kbytes for code).
3. Multitasking operating systems such as CP/M and XENIX can be implemented via the MMU.

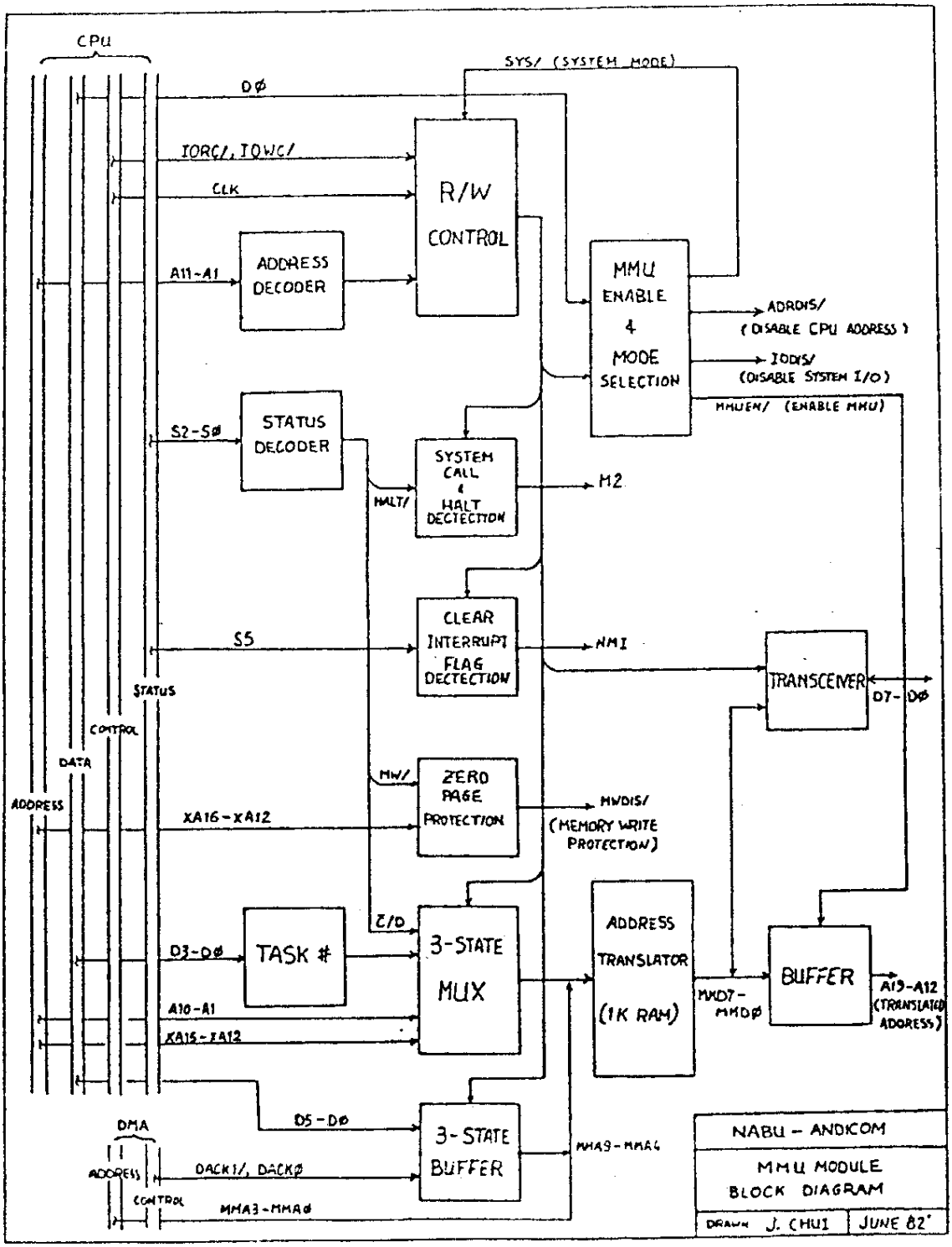

Figure 1 MMU Block Diagram

4.0 BLOCK DIAGRAM

Figure 1 is a block diagram of the NABU-1200 MMU. This section describes the function and use of each component comprising the block diagram.

4.1 Control Logic

The Control Logic section manages the MMU enable/disable function and I/O access.

4.2 3-State MUX

When the map is loaded to the Address Translator (RAM), the addresses (A10-A1) are passed through the 3-State Multiplexer. When the map has been loaded, the load addresses (A10-A1) are disabled.

4.3 3-State Buffers

The 3-State Buffers handle DMA access (diskette operations).

4.4 Address Translator

The Address Translator requires the following inputs and produces the A19-A12 translated addresses.

Inputs

1. Task number.
2. Code fetch/Data fetch signal.
3. Address bits A12-A16.

Output

Address bits A12-A19.

- all instruction fetches are through the code map
- if a^{19} is high & a memory fetch is requested then access is through the code map
- interrupt vectors are fetched through data map

5.0 OPERATIONS

The MMU is enabled via an I/O instruction which activates a one-way latch. When this latch is on, the MMU cannot be disabled until the system is reset.

The program that enables the MMU must reside in the same physical memory location, both before and after the MMU is activated. To ensure this, the instruction that activates the MMU must be in the logical 0th block of the program.

When the MMU is enabled, all memory access (including DMA cycles) is mapped accordingly. When the MMU is enabled, two modes of operation are possible:

1. SYSTEM.

2. TASK.

I/O operations are possible only in SYSTEM mode. The Address Translator is loaded via I/O instructions in SYSTEM mode only.

In TASK mode, the logical 0th block of data is write-protected.

The logical 0th block of data ~~and code~~ (Task and System) must be mapped at absolute address 00000H-00FFFH.

5.1 SYSTEM-To-TASK Transition

The instructions that control the transition from SYSTEM mode to TASK mode must reside in the logical 0th block of the program. These instructions perform the following functions:

1. Save the SYSTEM mode context.
2. Set the Task number.
3. Set the Jam signal (I/O instruction). — must be on in task mode or it an error
4. Set the MMU to TASK Mode (I/O instruction).
5. Load the Task information (except for the flags, code segment, and program counter).
6. Execute a Return from Interrupt (IRET) instruction.

5.2 TASK-To-SYSTEM Transition

TASK-to-SYSTEM transition is invoked by:

1. System Calls - via execution of a specific I/O and HALT instruction sequence.
2. All maskable interrupts.

The following operations are performed when the system is switched from TASK Mode to SYSTEM Mode:

1. The processor Interrupt Acknowledge signal sets the MMU to SYSTEM Mode. CS + PC are fetched via non-data table
2. The flags, code segment, and program counter are pushed onto the stack (because the Jam signal is still ON, the TASK data area is used, rather than the SYSTEM data area).
3. Control is passed to the corresponding interrupt handler.
4. The task context is saved in the TASK data area.
5. The Jam signal is reset.

5.3 Address Translator

The Address Translator can manage a maximum of 15 tasks, plus the system. Generally, when the System signal is OFF, the TASK area is used; when the System signal is ON, the SYSTEM area is used. Table 1 illustrates the logic used to determine the output.

System Signal	Jam Signal	Code/Data Fetch	Result
ON	ON	Code	System
ON	ON	Data	Task
ON	OFF	Code	System
ON	OFF	Data	System
OFF	ON	Code	Task
OFF	ON	Data	Task
OFF	OFF	Code	Error
OFF	OFF	Data	Error

Table 1 Address Translator Logic

6.0 PROGRAMMING INFORMATION

6.1 Address Translator Loading

The Address Translator is loaded and verified via I/O instructions (except in TASK Mode). Table 2 illustrates the Translator map assignment.

Example: the following instruction sequence maps the 0th page of system code to page 0 of physical memory.

```
MOV  DX,800H    ;get I/O address for page 0
MOV  AL,0       ;get page number
OUT  DX,AL      ;set translator map
```

each segment contains a bit address.
If owned by [illegible] memory from garbage

	CODE		DATA	
	Page No.	I/O Address	Page No.	I/O Address
System (Task 0)	0	800	0	840
	1	802	1	842
	2	804	2	844
	3	806	3	846
	4	808	4	848
	5	80A	5	84A
	6	80C	6	84C
	7	80E	7	84E
	8	810	8	850
	9	812	9	852
	10	814	10	854
	11	816	11	856
	12	818	12	858
	13	81A	13	85A
	14	81C	14	85C ← work/scratch
	15	81E	15	85E ← u-
	16	820	16	860 ← copy seg from
	17	822	17	862
	18	824	18	864 ← copy seg to
	19	826	19	866
	20	828	20	868
	21	82A	21	86A
	22	82C	22	86C
	23	82E	23	86E
	24	830	24	870
	25	832	25	872
	26	834	26	874
	27	836	27	876
	28	838	28	878
	29	83A	29	87A
	30	83C	30	87C
	31	83E	31	87E

Table 2 Translator Map Assignment

	CODE		DATA	
	Page No.	I/O Address	Page No.	I/O Address
Task 1 to system code 0	0	880	0	8C0 ← to system Data 0
	1	882	1	8C2
	2	884	2	8C4
	3	886	3	8C6
	4	888	4	8C8
	5	88A	5	8CA
	6	88C	6	8CC
	7	88E	7	8CE
	8	890	8	8D0
	9	892	9	8D2
	10	894	10	8D4
	11	896	11	8D6
	12	898	12	8D8
	13	89A	13	8DA
	14	~~8AC~~ 89C	14	8DC
	15	~~8AE~~ 89E	15	8DE
User code →	16	~~8B0~~ 8A0	16	8E0 ← User Data
	17	~~8B2~~ 8A2	17	8E2
	18	~~8B4~~ 8A4	18	8E4
	19	~~8B6~~ 8A6	19	8E6
	20	~~8B8~~ 8A8	20	8E8
	21	~~8BA~~ 8AA	21	8EA
	22	~~8BC~~ 8AC	22	8EC
	23	~~8BE~~ 8AE	23	8EE
	24	8B0	24	8F0
	25	8B2	25	8F2
	26	8B4	26	8F4
	27	8B6	27	8F6
	28	8B8	28	8F8
	29	8BA	29	8FA
	30	8BC	30	8FC
	31	8BE	31	8FE

Table 2 Translator Map Assignment

6.2 Command Description

Table 3 describes the instructions, data, and I/O addresses for the commands.

Command Function	I/O Address	Instruction	AL Content
Enable MMU	20H	OUT	(not relevant)
Set SYSTEM Mode	22H	OUT	00H
Set TASK Mode	22H	OUT	01H
Load Task Number	24H	OUT	(See Table 3.1)
Reset Jam Signal	26H	OUT	00H
Set Jam Signal	26H	OUT	01H
Clear System Call	26H	IN	(not relevant
Clear NMl	28H	IN	(not relevant)
Floppy disk I/O register	2AH	OUT	(See Table 3.2)
Hard disk I/O register	2CH	OUT	(See Table 3.2)
System Call	30H	OUT HLT	(not relevant)

Table 3 Command Description

	0	0	0	0	L3	L2	L1	L0
					v	v	v	v
System					0	0	0	0
Task 1					0	0	0	1
Task 2					0	0	1	0
Task 3					0	0	1	1
Task 4					0	1	0	0
Task 5					0	1	0	0
Task 6					0	1	1	0
Task 7					0	1	1	1
Task 8					1	0	0	0
Task 9					1	0	0	1
Task 10					1	0	1	0
Task 11					1	0	1	1
Task 12					1	1	0	0
Task 13					1	1	0	1
Task 14					1	1	1	0
Task 15					1	1	1	1

Table 3.1 AL Contents - Task Number

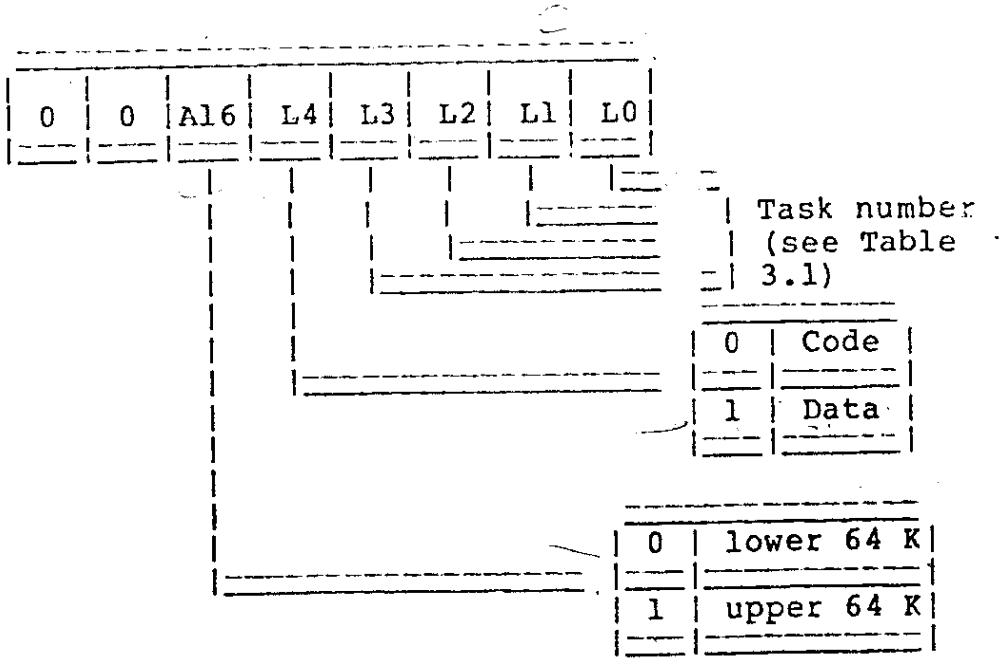

Table 3.2 AL Contents - DMA For Floppy/Hard Disk

6.3 Enable MMU

To enable the MMU, the following steps are performed.

1. Reset Jam Signal:

```
MOV   AL,0
OUT   26H,AL
```

2. Set Task number to System:

```
MOV   AL,0
OUT   24H,AL
```

3. Set SYSTEM Mode:

```
MOV   AL,0
OUT   22H,AL
```

4. Activate MMU:

```
OUT   20H,AL
```

6.4 SYSTEM-To-TASK

To pass control from SYSTEM Mode to TASK Mode, the following steps are performed:

1. Save SYSTEM Mode context:

```
PUSH  AX
PUSH  BX
PUSH  CX
PUSH  DX
PUSH  SI
PUSH  DI
PUSH  BP
PUSH  DS
PUSH  ES
PUSHF
MOV   system_sp_save,SP
MOV   system_ss_save,SS
```

2. Load stack information from task:

```
MOV   SP, task_sp_save
MOV   SS, task_ss_save
```

3. Load task number:

```
MOV   AL,task_number
OUT   24H,AL
```

4. Clear System Call and send EOI to Interrupt Controller:

```
IN    26H,AL    ;clear_system_call
MOV   AL,62H    ;send-EOI
OUT   42H,AL
```

5. Set Jam signal:

```
MOV   AL,1
OUT   2BH,AL
```

6. Set MMU to TASK Mode:

```
MOV   AL,1
OUT   22H,AL
```

7. Restore task context (excluding flags, which are automatically picked up on the IRET instruction):

```
POP   ES
POP   DS
POP   BP
POP   DI
POP   SI
POP   DX
POP   CX
POP   BX
POP   AX
```

8. Execute Return from Interrupt Instruction:

```
IRET
```

6.5 System Call Interrupt Handler

When a System Call occurs, the following steps are performed:

1. Save context of task making System Call (Task area):

```
PUSH    AX
PUSH    BX
PUSH    CX
PUSH    DX
PUSH    SI
PUSH    DI
PUSH    BP
PUSH    DS
PUSH    ES
```

Note that the flags need not be saved - System Calls are activated via interrupts - flags are saved during interrupt handling.

2. Check whether System Call is activated via proper System Call instructions:

```
IN   AL,20H   ;if bit 0 of AL set, then proper
              ;system call
```

an improper sys call is a HLT not preceded by OUT 30 (this traps to same location as system call)

3. Reset Jam Signal:

```
MOV  AL,0
OUT  26H,AL
```

4. Save stack information from task (System area):

```
MOV   task_sp_save,SP
MOV   task_ss_save,SS
```

The sequence out 30 20 usually causes one interrupt which is a proper system call. Sometimes two interrupts occur. The first [illegible] Bit attempts to put [illegible]

5. Restore system context:

```
MOV   SP,system_sp_save
MOV   SS,system_ss_save
POPF
POP   ES
POP   DS
POP   BP
POP   DI
POP   SI
POP   DX
POP   CX
POP   BX
POP   AX
```

5. Execute System Call.

6.6 Other Maskable Interrupt Handlers

To reset the interrupt, a context switch will be required in addition to the normal codes. Refer to the NABU-1200 Programming Documentation for details.

6.7 Non-Maskable Interrupt Handlers

Happens if user attempts to disable interrupts.

When a non-maskable interrupt occurs, the following steps are performed:

1. The flag word saved on the task's stack is modified so that the interrupt flag bit (bit 9 of the flag word) is set.

2. The NMI register in the MMU board is reset:

```
IN   AL,28H
```

3. The Return from Interrupt Instruction (IRET) is executed.

NMI does not put MMU into sys mode

7.0 PERFORMANCE

(To be announced).

8.0 INTERFACES

(To be announced).

9.0 PHYSICAL SPECIFICATIONS

9.1 Power Requirements

(To be announced).

9.2 Environmental Concerns

The MMU operates in the same environmental conditions as the CPU board.

9.3 Physical Size

(To be announced).

10.0 RELIABILITY AND SERVICING

(To be announced).

10.1 INSTALLATION AND MAINTENANCE REQUIREMENTS

(To be announced).

	CODE		DATA	
	Page No.	I/O Address	Page No.	I/O Address
Task 2	0	900	0	940
	1	902	1	942
	2	904	2	944
	3	906	3	946
	4	908	4	948
	5	90A	5	94A
	6	90C	6	94C
	7	90E	7	94E
	8	910	8	950
	9	912	9	952
	10	914	10	954
	11	916	11	956
	12	918	12	958
	13	91A	13	95A
	14	91C	14	95C
	15	91E	15	95E
	16	920	16	960
	17	922	17	962
	18	924	18	964
	19	926	19	966
	20	928	20	968
	21	92A	21	96A
	22	92C	22	96C
	23	92E	23	96E
	24	930	24	970
	25	932	25	972
	26	934	26	974
	27	936	27	976
	28	938	28	978
	29	93A	29	97A
	30	93C	30	97C
	31	93E	31	9FE

Floppy DMA

Table 2 Translator Map Assignment

	CODE		DATA	
	Page No.	I/O Address	Page No.	I/O Address
Task 3	0	980	0	9C0
	1	982	1	9C2
	2	984	2	9C4
	3	986	3	9C6
	4	988	4	9C8
	5	98A	5	9D0
	6	990	6	9
	29	9BA	29	.
	30	9BC	30	9FC
	31	9BE	31	9FE

Disc DMA

Table 2 Translator Map Assignment (Con't)

	CODE		DATA	
	Page No.	I/O Address	Page No.	I/O Address
Task 4	0	A00	0	A40
	1	A02	1	A42
	2	A04	2	A44
	.	.	.	.
	.	.	.	.
	30	A3C	30	A7C
	31	A3E	31	A7E
Task 5	0	A80	0	AC0
	1	A82	1	AC2
	2	A84	2	AC4
	.	.	.	.
	.	.	.	.
	30		30	
	31		31	
Task 6	0	B00	0	
	1	B02	1	
	2	B04	2	
	.	.	.	.
	.	.	.	.
	30	B	30	B7C
	31	B40	31	B7E
Task 7	0	B80	0	
	1	B82	1	
	2	B84	2	
	.	.	.	
	.	.	.	
	30			BFC
	31			BFE

Table 2 Translator Map Assignment (Con't.)