EMACS IOP

Don North — hardware for opus.

PROCESSOR

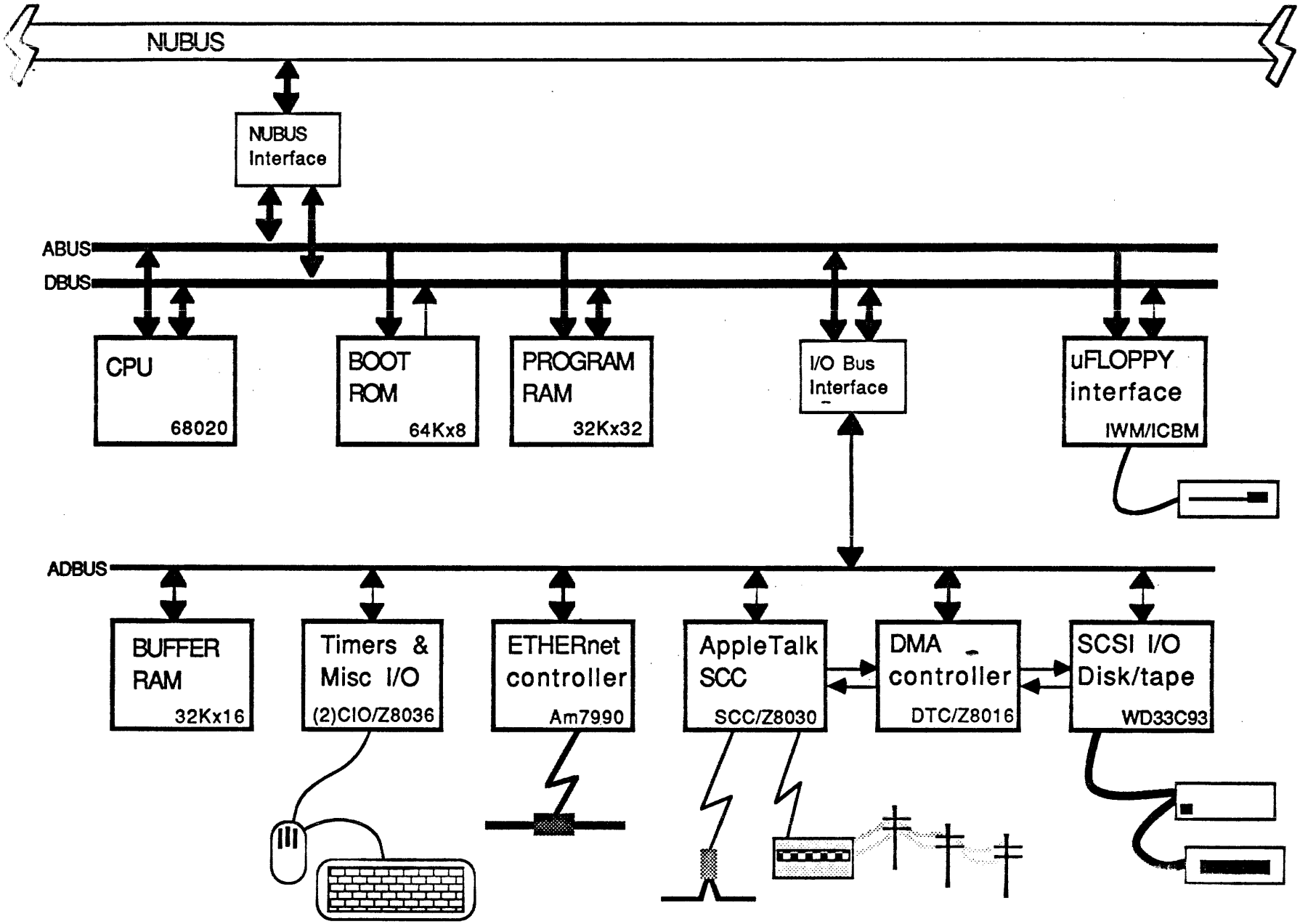
- MC68020, 12 MHz [16 MHz]
  [FPU & PMMU/MMB slots in CPU mode]

MEMORY

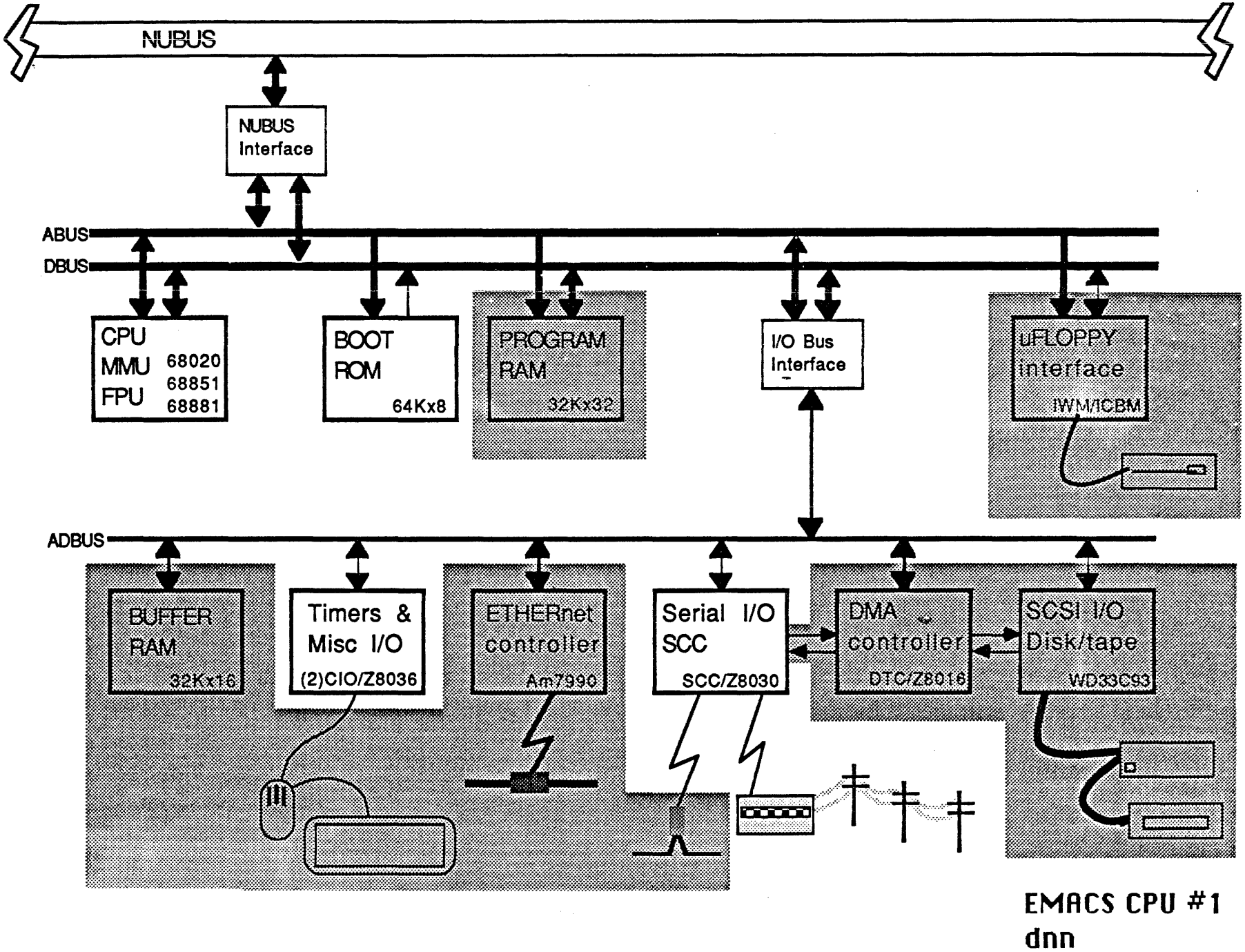
- Boot ROM, 64Kx8
- Program RAM, 32Kx32
- I/O Buffer RAM, 32Kx16

I/O

- NUBUS 32-bit master/slave interface
- uFLOPPY Disk Interface (IWM/ICBM ala MAC)
- Two channel DMA controller (Z8016 DTC)
  Byte/word assembly, chained operations
- AppleTalk/RS232 Serial Interface (Z8030 SCC)
  'A channel' DMA to/from Buffer RAM via DTC at 1.33 MB/s
- High performance SCSI Interface (WD33C93)
  DMA to/from Buffer RAM via DTC at 2.66 MB/s
- ETHERnet Interface (Am7990 LANCE)
  DMA to/from Buffer RAM at 3.33 MB/s
- Timers and Misc I/O Interface (2xZ8036 CIO)
  'Front Desk Bus' port to mouse, keyboard, ...
  MAC real-time clock (T.O.D., 1 s interrupt)
  Six 16-bit timers (1 us - 1 s resolution, cascadable)
- Common features
  'Direct addressing' of control registers (SCC, CIOs, SCSI)
  Fully vectored interrupts (SCC, CIOs, DTC)
  Single-vector interrupts (SCSI, LANCE)
  Separate I/O bus for parallel CPU-I/O DMA operations



EMACS IOP
dnn



EMACS CPU #1
dnn

EMACS

... more than just another text editor

OVERVIEW

EMACS (Experimental **MAC**intosh **S**ystem) is a high-performance Motorola MC68030 CPU, MC68881 floating point coprocessor, 1MB-to-8MB dynamic RAM subsystem, NuBus interface, and a little bit of local I/O, all on a single Apple-format NuBus module. It is designed to be a 'plug-in' application accelerator for Macintosh II (or other...) NuBus-based systems, using either the MacOS, OPUS, or A/UX operating systems. Projected performance is about 2X that of the existing Macintosh II on compute-bound benchmarks

CPU SUBSYSTEM

The CPU subsystem consists of an MC68030 CPU/MMU and MC68881 (or '882) floating point coprocessor; support for the Weitek FP daughtercard 'super-socket' (a superset of the '881/2) has also been provided (but is currently untested). Depending upon the availability of parts, and the access time of the RAMs used in the memory subsystem, any clock rate from 12.5MHz thru 25MHz can be supported. Note that the CPU subsystem clock runs asynchronously to the 10MHz NuBus clock; this was done to allow a wide range of CPU clock rates to be employed in this prototype configuration.

MEMORY SUBSYSTEM

RAM

The memory subsystem employs dynamic RAMs, using either four or eight 256Kx8, 1Mx8, or 4Mx8 SIMM memory modules, for total capacities of 2, 8, or 32 megabytes respectively (mixing and matching are also possible). Depending upon the processor clock rate, either 120ns, 100ns, or 80ns memory modules may be required for 'best' performance. The current system timing has been tuned for the use of 100ns RAMs and a 25MHz processor subsystem.

The MC68030 MPU has a *burst-mode* memory cycle used to fill its internal instruction and data caches. In this access mode, four longwords are requested (in a circular buffer fashion, starting with the first longword required) within one sixteen-byte aligned block of memory. The EMACS memory controller, in conjunction with fast-page-mode DRAMs, supports these burst-mode accesses employing either 1 or 2-wait state processor cycles, by a clock-stretching technique (normally wait-states only occur in pairs). The following table gives further details concerning the performance of the memory subsystem, and a comparison to the existing MacintoshII memory interface.

cycle type	CPU clock	memory speed	wait states	effective ns/4bytes	computation...
single read	16.7MHz	120/60ns	4	240	=8*30/1
...	20.0MHz	100/50ns	4	200	=8*25/1
...	25.0MHz	100/50ns	6	200	=10*20/1
...	25.0MHz	80/40ns	4	160	=8*20/1
single write	16.7MHz	120/60ns	2	180	=6*30/1
...	20.0MHz	100/50ns	2	150	=6*25/1
...	25.0MHz	100/50ns	3	140	=7*20/1
...	25.0MHz	80/40ns	2	120	=6*20/1
burst read	16.7MHz	120/60ns	4/1	128	=17*30/4
...	20.0MHz	100/50ns	4/1	106	=17*25/4
...	25.0MHz	100/50ns	5/2	110	=22*20/4
...	25.0MHz	80/40ns	4/1	85	=17*20/4
MacII cycle	15.7MHz	120/60ns	6[1]	320	=10*32/1

[1] the '020 has a basic 6-state CPU cycle compared to 4-states in the '030; 2 has been added to these entries to make the meaning of the column consistent

Dynamic RAM refresh is performed automatically, stealing cycles when necessary, or hiding them behind NuBus or I/O accesses if possible. CAS-before-RAS refresh is used; a request is generated at approximately 15.2 microsecond intervals.

ROM

A small 64Kx8 ROM is resident to iniitialize the processor during the power-up sequence, and provide configuration information to the NuBus host as well. It is not expected this ROM will hold any significant amount of code during run-time, as accesses to the RAM (because of burst-mode operation) are *significantly* faster. In future versions, a much larger (slower) ROM could be used to hold proprietary code/data, prior to copying it into RAM for normal use. The ROM is thus a module-resident 'floppy' containing Apple-unique system software.

NUBUS SUBSYSTEM

OVERVIEW

The NuBus interface is a full 32 bit master/slave port that supports all the transfer size and byte alignment combinations of the '030 processor (ie, unaligned transfers are handled correctly in hardware). Block mode transfers are not supported by the interface; they are not generated on the NuBus, and block mode transfers directed at the card are not 'recognized' (they return an

immediate acknowledge / early termination to the sender). In actuality, the '030's concept of a block transfer does not mesh well with that of the NuBus. The '030 starts with the longword it desires, and wraps around within the block; the NuBus always starts at the beginning of the aligned block.

Both the normal slot space addresses 0xFNxxxxxx (for I/O accesses) and the 'super' slot address space 0xNxxxxxxx (for RAM accesses), where N is the NuBus SlotID, are decoded for slave accesses to the card.

LOCKED ACCESSES

Locked accesses (ref NuBus specification, '030 *TAS,CAS,CAS2* instructions), required for multi-processor inter-communication via shared memory, are handled correctly in hardware.

Outgoing read-modify-write interlocked sequences initiated by the '030, directed at the NuBus, will perform the appropriate resource 'lock' and 'unlock' attention cycles, and all interim NuBus transactions, without releasing the NuBus to another master. The slave device must recognize (or possibly ignore) these extra attention cycles, and lock/unlock local resources as appropriate.

Incoming resource 'lock/unlock' attention cycles are recognized by the NuBus interface, and cause the '030 to be stalled for the duration of the interlocked sequence. Normally, the '030, after being stalled by an externally generated NuBus access, would restart immediately after the external access was complete.

ERRORS

The NuBus interface does not ever generate the 'go-away, try-again-later' response to an external NuBus cycle. In reaction to this response being generated by another NuBus device (ie, the MacII) to a locally generated NuBus transaction, the NuBus interface will terminate the current NuBus cycle, re-arbitrate the NuBus, and immediately re-try the cycle (ad nauseum...), transparent to the local '030 bus cycle. It just takes a lot longer, sometimes.

The only time the NuBus interface will return the 'buss-error' condition is when an actual error condition is signaled by the slave, or a NuBus timeout is detected by the MacII motherboard.

DEADLOCK

Since the '030 and its local bus operate in parallel with the NuBus, a deadlock situation can sometimes occur when the '030 desires to access the NuBus, but the current NuBus master is attempting to get access to the local '030 bus... instant hang-up unless one or the other gives up.

The 'abort and retry' function of the '030 is employed in these situations; it will surrender the local bus, and later retry the aborted cycle when the external transaction is complete (transparent to software). The interlocked memory instructions (TAS, CAS, CAS2) will operate correctly in this environment, *except* for CAS2 when the first operand address is in local memory and the second is in external NuBus memory. In this configuration, a buss-error may result to *either* the processor executing the locked sequence, or the *unsuspecting* current NuBus master, even though nothing is apparently amiss. This arises because the '030 will not arbitrate its local bus once it has entered a locked transaction sequence; deadlock occurs until the NuBus timeout logic aborts the current master, and allows the '030 processor executing the locked sequence to become NuBus master. If the operands to CAS2 are guaranteed to be in the same memory (either locally or on NuBus) this situation will never occur.

Note that while the local bus is temporarily given to the external requestor, the pending request for the NuBus generated from the aborted local cycle is still present; thus the '030 keeps its place in the queue for an upcoming NuBus slot. This increases performance significantly at the expense of software complexity. If the retry cycle subsequently aborts (buss-errors), the error handling routine must execute at least one NuBus cycle (eg, read location 0x0) to clear the pending request. Until this is done, the NuBus will be hung waiting for the pending request to complete.

I/O SUBSYSTEM

OVERVIEW

Resident on each EMACS module is a Motorola MC68901 'Multi-Function Peripheral', hereafter referred to as the MFP. It contains an eight-bit parallel I/O port (used for status/control bits), four eight-bit timers, an interrupt controller, and a simple USART. For much more detailed information on the MFP itself, refer to the Motorola MC68901 Programmer's Manual.

MFP TIMERS/USART DEFINITIONS

The MFP contains four eight bit timers that can be used to generate interrupts at predetermined intervals. The timers can be used individually, clocked by the (possibly internally) prescaled 3.6864MHz external oscillator, or in a daisy-chained fashion, to build a 16 bit or 24 bit capable timer. Externally, the timers are connected such that 'C' -> 'B' -> 'A', so that timer 'C' would provide the least significant counter bits, 'B' the middle, and 'A' the most significant.

Timer 'D' can be used as a software-defined timer, or as the baud-rate generator in conjunction with the simple on-board serial I/O port. A MacPlus DIN-8 style connector out the back provides access to this serial port (RS232-C compatible only). The magic frequency 3.6864MHz was chosen for the external input to easily generate the standard 19.2K/9600/... 16X baud rate clocks accurately.

MFP INTERRUPTS

All interrupts are generated by the MFP, and occur at level '6' to the '030 processor. The sixteen internal sources of interrupts are detailed in the MFP Programmer's Manual, but include the 8 I/O port bits, the 4 timers, and 4 USART related sources (not all are used in EMACS). An internal vector base register provides vectoring of the highest priority interrupt source to one-of-sixteen different interrupt vectors.

MFP I/O PORT DEFINITIONS

The following table presents the bit definitions of the eight-bit MFP 'iop' (I/O port) register.

```
MFPiop<07>              (interrupt input, falling)                    NMI~
          Asserted when the interrupt button is pushed, or when ANY
          access is made to an MFP address with A<05>=1 (normally 0)
          Not 'debounced'.

MFPiop<06>              (interrupt input, falling)                FPU_INTR~
          Asserted by the WEITEK floating point unit when it desires
          service for some exception condition.

MFPiop<05>                    (output, high)                        MEM_SIZE
          Memory size / bank select control.  Allows memory to
          appear at contiguous locations in physical address space.
               LOW = use A<22> (w/ 1Mx8 SIMMs)
              HIGH = use A<24> (w/ 4Mx8 SIMMs)

MFPiop<04>                    (output, low)                     PROG_RESET~
          When asserted, its falling edge causes a 10 microsecond
          delay, and then a 1500 microsecond CPU, FPU, and MFP
          'hard' external reset to be generated.  Normally
          deasserted (pulled high) by an external pullup resistor.

MFPiop<03>                    (output, high)                        POWER_UP
          Asserted during the PowerUp/RESET sequence by an external
          active-high pullup; cleared under program control.  When
          ASSERTED causes ROM to be mapped at address 0x0, NuBus
          accesses to be disabled, and 24 bit physical address mode.
          See the section ADDRESS SPACE ALLOCATION / RESET-MODE.

MFPiop<02>                    (output, low)                         HP_INTR~
          When asserted, causes a NuBus non-master request (NMRQ~)
          to be sent to the MacII.

MFPiop<01>                  (data input, high)                      SLOT_BIT
          This 'serial' input is used to read the NuBus SlotID bits
          during the reset sequence (when PowerUp=1).  See the
          section ADDRESS SPACE ALLOCATION / RESET-MODE.

MFPiop<00>              (interrupt input, falling)                   IP_INTR~
          Asserted when the IPI - InterProcessorInterrupt - address
          is referenced.
```

ADDRESS SPACE ALLOCATION

RESET-MODE RAM/ROM DECODE

At power-up (or after a reset occurs) the '030 processor fetches its initial SP and PC from physical address locations 0x0 and 0x4 (both are longwords). These addresses would normally be external NuBus resources (ie, of RAM memory in a MacII configuration), which is unacceptable. Thus there is a PowerUp configuration bit (ref the MFP) that selects between the two separate modes of address space decode, 'initial' and 'normal'.

In 'initial mode', the upper eight bits of the physical address are effectively ignored, the processor is operating in 'physical 24 bit address' mode. All external accesses to NuBus are disabled; only on-card I/O resources can be selected (MFP, IPI, ROM). On-card RAM is NOT accessible. In effect the CPU behaves as if A<31:24>=0xF4, independent of what it *really* is.

Determining the NuBus SlotID must be done in this configuration, as address bits <25:24> are the selector for the SlotID input bit in the MFPiop register (ref the MFP). Thus reading the MFP I/O port with A<25:24>=n will return the card's NuBus SlotID bit<n> (where n=0,1,2,3) in the SlotID bit position of the MFPiop data register. A hack, but it doesn't need to be done often.

In 'normal mode', the upper eight bits of the physical address are decoded; the processor is operating in 'physical 32 bit address' mode. External accesses to NuBus are possible, on-card I/O resources can be selected (MFP, IPI, ROM), as well as on-card RAM. Details follow.

NORMAL-MODE RAM/ROM DECODE

With the PowerUp bit in the MFPiop register set to 'normal' mode, full 32 bit physical address operation, and external NuBus accesses, are possible.

The upper eight bits of the physical 32 bit address are now decoded, and generate either an on-card or NuBus access, as appropriate. The special addresses 0x[4x/F4]xxxxxx always decode to match the current card's SlotID, and never go off card. Most all other addresses will go off card onto the NuBus, except for those that match the current slot id N, and are of the form 0xFNxxxxxx (normal slot space; I/O and ROM) and 0xNxxxxxxx (super slot space; RAM). These, of course, stay on-card. The value of the NuBus SlotID N ranges from 0x9 thru 0xE in the MacintoshII implementation. The EMACS card supports SlotIDs in the 0x7-0xE (8 slots MAX) range.

In 'normal' mode, RAM is mapped starting at on-card location 0xS0000000 (bank 0) and 0xS0400000 (bank 1, MEM_SIZE=0) or 0xS1000000 (bank 1, MEM_SIZE=1), where S is the on-card slot select pattern '4' or 'N'. Note that when 2 banks of 4 256Kx8 SIMM RAMs are used (not likely, however), the memory will be physically discontiguous on card, as the banks are located on 4MB or 16MBboundaries only. The on-chip MMU can be used to 'fix' this, if desired.

I/O SPACE DECODE

The I/O space peripherals are the MFP, IPI, and ROM.

The MFP actually contains 24 internal registers, which are addressed as offsets to the base address, shifted left by 6 positions. For example, register 0x12 would be addressed at location 0xFS400000+(0x12<<6), or 0xFS400480. All the registers are 1 byte wide, and only single byte read/write accesses should be performed.

The ROM is 64Kbytes and is also only 1 byte wide; but is connected in such a way that the dynamic bus sizing capability of the '030 can be employed to access bytes, words, or longs transparently to the software. Writes to the ROM do no harm, and are effectively ignored. Valid ROM addresses are from 0xFS000000 thru 0xFS3FFFFF and from 0xFSC00000 thru 0xFSFFFFFF (aliased). ROM resides at both the 'top' of the 16 MByte slot address space 0xFNxxxxxx as required by the MacII OS Slot manager; and at the 'bottom' as required by the CPU during reset.

The IPI - InterProcessorInterrupt - facility is accessed at address 0xFS800000. Any type access is legal (byte/word/long, read/write) - only the address is important; no data is transferred. Accesses

to this address cause transitions to occur on selected MFP interrupt input lines, and the corresponding interrupt to be posted. See the MFP and INTERRUPT sections for further info.

Any access to an MFP I/O port address with A<05>=1 also triggers the programmer interrupt button port bit. See the MFP port bit definition table for further information.

ALLOCATION

The address space of the EMACS processor, viewed from the inside looking out, is described in the following table. Two major sections are defined, one for RAM and another for I/O. Each major section has two subparts; the first details the address pattern necessary for selection of that major group, the second itemizes the subselects for individual elements within the group.

```
          33222222 22221111 1111110000000000
          10987654 32109876 5432109876543210
========  ======== ======== ================  ============================
RAM:      0100.... ........ ................  PowerUp=0 & A<31:24>='4X'
          ssss.... ........ ................  PowerUp=0 & A<31:24>='sX'
          -------- -------- ----------------  --
          ......00 0b****** **************..  RAM, MEM_SIZE=0
          ......0b ******** **************..  RAM, MEM_SIZE=1
========  ======== ======== ================  ============================
I/O:      XXXXXXXX ........ ................  PowerUp=1
          11110100 ........ ................  PowerUp=0 & A<31:24>='F4'
          1111ssss ........ ................  PowerUp=0 & A<31:24>='Fs'
          -------- -------- ----------------  --
          ........ 00...... ****************  ROM (64Kx8)
          ........ 01...... .....*****0.....  MFP (32x8)
          ........ 10...... ................  IPI
          ........ 11...... ****************  ROM (64Kx8)
========  ======== ======== ================  ============================
          33222222 22221111 1111110000000000
          10987654 32109876 5432109876543210
```

DISCLAIMER

Any errors or unclear points? Let me know. Don.