Inter-Office Memorandum

To Distribution

Date July 7, 1977

From Jarvis

Location Palo Alto

Subject OIS Processor/Peripheral Controller Interface Guidelines

Organization SDD

XEROX

Filed on: <Jarvis>controller-guides.bravo

This memo proposes guidelines to use when designing the interface between an OIS processor executing the Mesa instruction set, hereafter called the emulater, and a peripheral device controller, hereafter called the controller. This memo also outlines the style of Pilot i/o transactions.

The Controller Status Block

The processor architecture allows as many as sixteen controllers attached to a single system element. The architecture assigns each controller a block of sixteen registers in the i/o page, virtual memory page zero. This register block, the controller status block, provides status information for the emulator. The controller in turn periodically polls the CSB for commands from the emulator. For some peripherals, polling the CSB is an inappropriate method to initiate action. In that case, the emulator can use the OUTPUT instruction to initiate action directly. The emulator reserves the first CSB, virtual memory locations 0-15, for the processor and integrated controllers and the last CSB, locations 240-255, for handling faults.

```
IOPage: ARRAY [0..15] OF ControllerStatusBlock;

IOCBHandle: CARDINAL;

ControllerStatusBlock: TYPE = MACHINE DEPENDENT RECORD[
  chain: IOCBHandle,
  wakeUpMask: CARDINAL,
  deviceDependent: ARRAY [0..13] OF UNSPECIFIED]; -- CSB is 16 words
```

The first word of the CSB points to a chain of input/output control blocks, IOCB's. The current plans are for Pilot to allocate all IOCB's from the first 64k of virtual memory. Using 16 instead of 24 bit pointers reduces the size of IOCB's and alleviates timing problems since a double word store is unnecessary to write IOCB pointers. It would seem reasonable to maintain other Pilot data structures (e.g. ECB's and channels) in the first 64k of virtual memory for the same reasons. As the controller chases down the chain, it updates the first word of the CSB to indicate the currently active IOCB. When the controller finds a zero, it stops processing and stores the zero into the CSB.

The second word of the CSB contains the wake up request bit mask. The controller looks here to determine which process or processes to wake up when events require.

The other 14 words of the CSB are unassigned and can be used for device dependent status and commands.

Input/Output Control Blocks

IOCB's hold information passed between the emulator to control input and output transactions. The emulator places a command and buffer descriptor in the IOCB and puts the IOCB on the chain. After the controller processes the IOCB, it stores a completion code, and, depending upon the command code, generates process wake ups.

```
IOCB: TYPE = MACHINE DEPENDENT RECORD[
  next: IOCBHandle,
  buffer: LONG POINTER TO UNSPECIFIED,
  command: IOCBCommandCode,
  completion: IOCBCompletionCode,
  byteCount: CARDINAL,
  maxByteCount: CARDINAL,
  synch: EventID,
  previous: IOCBHandle,
  deviceDependent: ARRAY OF UNSPECIFIED];
```

The first word of an IOCB points to the next IOCB on the chain. A zero indicates the end of the chain.

The next two words are a 24 bit pointer to the buffer.

The emulator writes a command code into the fourth word. Three bits of this word are reserved to control process wake ups.

```
IOCBCommandCode: TYPE = MACHINE DEPENDENT RECORD[
  wakeUpAlways,                -- bit 0
  wakeUpOnError,               -- bit 1
  stopOnError: BOOLEAN, -- bit 2 Don't process next IOCB on error
  deviceDependent: [0..17777B]];
```

The controller wakes up the processes indicated by the wake up request bit mask in the CSB.

The controller writes a completion code in fifth word. Two bits of this word are reserved.

```
IOCBCompletionCode: TYPE = MACHINE DEPENDENT RECORD[
  processed:                   -- bit 0
  error:                       -- bit 1
  deviceDependent: [0..37777B]];
```

The rest of the word contains a device dependent code. The emulator can determine whether the controller has processed the IOCB by writing zero in this word before placing the IOCB on the chain. The controller guarantees to write a non-zero quantity after processing.

The sixth word of the IOCB holds a count of valid data bytes in the buffer. If the controller fills the buffer, then this word holds the number of bytes written after processing the IOCB. If the controller empties the buffer, this word contains the number of bytes to transfer.

The emulator uses the seventh word of the IOCB to write the maximum length of the buffer. If the controller fills the buffer, the number of bytes written must not exceed the maximum length. If the controller empties the buffer, this word is redundant with the byte count.

The eighth word holds a Pilot event ID. On completion of processing the IOCB, the controller posts the event. This has the same effect as a Mesa call to post the event, but this method obviates a process switch.

The ninth word contains a pointer to the previous IOCB in the chain. This word facilitates insertion of IOCB's in the middle of the chain. Not all devices require such flexible controll over the IOCB; in that case, this field is omitted.

The last part of the IOCB can be used to hold device dependent status and control. For example, the floppy disk uses this area to hold the disk block header and label.

c: Crowther,
Lauer,
Lynch,
Metcalfe,
Ogus,
Rosen,
Schwartz,
Shultz,
Stottlemyre,
Thacker,