A Massbus Mystery, or,
Why Primary Sources Matter, Even In Computer History

Bob Supnik, 24-Sep-2004

Summary

In preparing a simulator for the VAX-11/780, I discovered that all the extant printed documentation for the DEC RP04/RP05/RP06 controllers is incorrect. Further, VMS followed this error in its drivers, creating a latent bug that has been present since the first release of the operating system in 1977.

Background: The Massbus

The Massbus is a simple, 16b, high-speed interconnect between a CPU host adapter and one or more mass storage devices. DEC created the Massbus in the early 1970's, to provide a CPU-to-mass-storage interconnect that was faster than the Unibus. The Massbus was implemented in the PDP-11/70 (via the RH70 host bus adapter) and the DECsystem-20 (via the RH20 host bus adapter). The Massbus was the primary storage interconnect on the VAX-11/780 (via the RH780 host bus adapter). Massbus storage could also be connected to Unibus PDP-11's (via the RH11 host bus adapter).

The Massbus implemented a very simple command and control structure between the host bus adapter and devices. The host adapter maintained the address and word count (DMA) logic. It communicated with the devices via register reads and writes. The host adapter mapped host addresses either to internal (adapter) registers offsets, or to external (device) register offsets. On the PDP-11, this mapping was quite complicated, and a mapping PROM was used between host addresses and register offsets; on the VAX, it was very simple, with different partitions of the adapter's address space being used for internal offsets and external offsets.

RP vs RM, VAX vs PDP-11

The issue at hand arose in trying to understand how this mapping actually worked across different Massbus storage devices, particularly the RP and RM families of removable disk packs. The RP04/05/06 family came first, based on buyout Memorex drives; the RM03/RM05/RP07 family (yes, the RP07 was an RM, despite the name) came later, based on buyout CDC and ISS drives. According to the maintenance manuals for the respective drive families [1,2], the internal register offsets were not quite the same:

Offset$_{10}$	RP family	RM family
0	CS0	CS0
1	DS	DS

2	ER1	ER1
3	MR	MR
4	AS	AS
5	DA	DA
6	DT	DT
7	LA	LA
8	ER2	SN
9	OF	OF
10	DC	DC
11	CC	HR
12	SN	MR2
13	ER3	ER2
14	EC1	EC1
15	EC2	EC2

Because the RH780 didn't map the external registers in any way, this difference was also reflected in the VMS driver.

But the PDP-11 (RH70/RH11), which did map the registers, showed a different picture:

Address$_8$	RP family	RM family
176700	CS0	CS0
176702	RH BA	RH BA
176704	RH WC	RH WC
176706	DA	DA
176710	RH CS2	RH CS2
176712	DS	DS
176714	ER1	ER1
176716	AS	AS
176720	LA	LA
176722	RH DB	RH DB
176724	MR	MR
176726	DT	DT
176730	SN	SN
176732	OF	OF
176734	DC	DC
176736	CC	HR
176740	ER2	MR2
176742	ER3	ER2
176744	EC1	EC1
175746	EC2	EC2

The correspondence between RP and RM registers in the VAX and in the PDP-11 is identical, except for RP SN, RM SN, RP ER2, and RM HR. If the VAX offsets were correct, then somehow the RH70/RH11 was mapping 176730 to offset 12 on the RP and offset 8 on the RM, and 176740 to offset 8 on the RP and offset 12 on the RM. How could this be?

First Hypothesis: Magic In the RH70/RH11

My first hypothesis was that, somehow, the RH70/RH11 was generating different mappings for the RP and RM drive families. Because this mapping was done with a PROM [3], this hypothesis implied that the RH70/RH11 had to be customized for the drive type. Further, RP and RM drives could not be mixed on the same PDP-11 Massbus controller, because addresses 176730 and 176740 would be incorrectly mapped if the drive type and controller PROM didn't match.

A Beautiful Theory vs Ugly Facts

This hypothesis was quickly overwhelmed by evidence from developers and users. A typical response was from Paul Koning, from RSTS/E development.

> “I'm somewhat puzzled about all this but I know for a fact that we supported mixed configs, and we had all sort of odd mixes on our development machines. Remember DECnet host ARK? It was called that because it had "two of everything". (Eventually that was more than was possible, but it had an amazing collection even so. I distinctly remember RP04, RP06, RM03, and RP07 disks mixed on the two RH70s.)”

A perusal of the RSTS/E driver showed that of the suspect registers, SN and MR2 were never accessed, and ER2 was only accessed if the drive was known to be an RP. So even with scrambled numbering, mixed strings would work, provided the RH70/RH11 always used an RP-style mapping.

TOPS-10 and TOPS-20 developers were even firmer: mixed configurations were not only supported but routine. The only restriction was that disks and tapes could not be mixed on the same Massbus adapter. The TOPS-10 and TOPS-20 drivers accessed SN and ER2 and expected them to be in their proper Unibus locations.

Further, the TOPS-10 and TOPS-20 drivers for the RH20 (which, like the RH780 on the VAX, did not map device offsets) stated that the RP offset for SN was 8, not 12, and for ER2 was 12, not 8.

TOPS-10 and TOPS-20 ran with real hardware; so too did VMS. Who was right?

Back To The Primary Source

At this point, the only remaining option was to consult the primary source for computer designs: the schematics. Fortunately, the schematics for the

RP04/RP05/RP06 were online, in Al Kossow’s invaluable archive. The schematics provided the answer.

The RP04/05/06 implemented register decoding with a 74154 4:16 demultiplexor. The selects were laid out in numeric order, with 8 = SN and 12 = ER2 [4]. There were no jumpers or select swizzling logic, before or after the demultiplexor. The PDP-11, TOPS-10, and TOPS-20 were right. The RH70/RH11 needed only one, consistent mapping between host addresses and Massbus offsets. The RP maintenance manual, and the VMS driver, were wrong.

The Smoking Gun; And An Explanation

So how did VMS work? The answer couldn’t be simpler: although the register offsets for SN and ER2 were defined, they were never used. It didn’t matter that they were wrong; it was only a problem in the comments, not in functional operation. The definitions were probably copied over on “day 1” from an incorrect document (like the maintenance manual) and never changed.

As confirmation, the VMS error logging facility (ERF) differed from the driver. The error logger stored the RP registers in numeric order, lowest to highest, and then defined the following data structure to access the resulting information (I’ve added the implicit register offsets to make the correspondence clearer):

```
{
{ RP04/5/6/7 Disk Device Error Sub-packets
{

Aggregate RP0X_DE_PKT structure prefix RP0X_DE$;
    MBA_REGS structure longword unsigned dimension 7; /* MBA adapter registers
        MBA_CNFG longword unsigned;         /* Configuration register (RH780)
        MBA_CNTRL longword unsigned;        /* Control register
        MBA_STAT longword unsigned;         /* Status register
        MBA_VA longword unsigned;           /* Virtual address register
        MBA_BYTE_CNT longword unsigned;     /* Byte count register
        MBA_FNL_MAP longword unsigned;      /* Final map register
        MBA_PRE_MAP longword unsigned;      /* Previous map register
    End MBA_REGS;
0    CSR1 longword unsigned;                /* RP04/5/6/7 control/status reg.
1    DRV_STAT longword unsigned;                    /* RP04/5/6/7 drive status register
2    ERROR1 longword unsigned;              /* RP04/5/6/7 error register
3    MAINT longword unsigned;               /* RP04/5/6/7 maintenance register
4    ATTN_SUM longword unsigned;                    /* RP04/5/6/7 attention summary reg.
5    D_ADDR longword unsigned;              /* RP04/5/6/7 desired address reg.
6    DRV_TYP longword unsigned;             /* RP04/5/6/7 drive type register
7    LOOK_AHEAD longword unsigned;          /* RP04/5/6/7 look ahead register
8    SN longword unsigned;                         /* RP04/5/6/7 serial number register
9    OFFSET longword unsigned;              /* RP04/5/6/7 offset register
10   D_CYL longword unsigned;               /* RP04/5/6/7 desired cylinder addr.
11   CUR_CYL longword unsigned;             /* RP04/5/6/7 current cylinder addr.
12   ERROR2 longword unsigned;              /* RP04/5/6/7 error register 2
13   ERROR3 longword unsigned;              /* RP04/5/6/7 error register 3
14   ECC1 longword unsigned;                /* RP04/5/6/7 ECC position register
15   ECC2 longword unsigned;                /* RP04/5/6/7 ECC pattern register
End RP0X_DE_PKT;
```

The error logger, which certainly did care about the definitions of SN and ER2, had them in the correct (i.e., schematic) order.

Conclusions

In an article on SIMH in ACM Queue, I wrote,

> "As in most forms of historical research, primary sources (schematics, microcode listings, and maintenance documentation) are best; secondary sources such as handbooks, marketing material, textbooks, and even user manuals cannot be trusted." [5]

As this Massbus mystery illustrates, the definition of primary sources has to be narrowed further: even maintenance manuals cannot be trusted. Errors can pile on errors over time: user manuals from maintenance manuals, drivers from user manuals, etc. And only reference to the schematics can unravel a 25+ year old error.

Acknowledgements

Paul Koning, Fred Van Kempen, Mark Crispin, Dave Carroll, and Joe Smith all provided personal evidence that knocked down my first hypothesis and eventually led me to look at the RP schematics. Andy Goldstein confirmed my analysis of the VMS drivers. Al Kossow's multi-year project to scan schematics and manuals and put them online made the entire effort possible.

References

[1] Digital Equipment Corporation, "RP05/RP06 Device Control Logic Maintenance Manual", EK-RP056-MM-01, December 1975

[2] Digital Equipment Corporation, "RM Massbus Adapter Technical Description Manual", EK-RMADA-TD-001, October, 1980

[3] Digital Equipment Corporation, "RH11B Field Maintenance Print Set", MP00382, schematic BTCA, Bus Control page 1

[4] Digital Equipment Corporation, "RP04/05/06 Field Maintenance Print Set", MP-00086, schematic RG5, Register Logic page 5

[5] Bob Supnik, "Simulators: Virtual Machines of the Past (and Future)", ACM Queue, Vol 2 No 5, July/August 2004