The Single-Board 6502

Eric Rehnke

High-Speed Data Transfer

Necessity is INDEED the mother of invention.

For quite some time I've thought about how neat it would be to have some way of transferring data at high speed between two computers. But, as usual, there was always something "more important" to do.

Recently, the need arose to have such a high-speed data transfer system.

As newsletter editor for INTERACTIVE (a newsletter published by Rockwell for the AIM 65), I frequently need to print AIM 65 program listings.

Now the AIM is a great little machine, and the on-board thermal printer is very convenient but a 20 column wide assembly language or BASIC listing just doesn't cut it for publication.

Hooking my Decwriter up to the AIM wouldn't solve the problem because AIM's ROM assembler still formats the outut for a 20 column wide printout.

Clearly, the only practical solution was to somehow move the source code over to my KIM system and assemble it with the HDE assembler.

Fortunately, except for the fact that AIM 65 text editor doesn't use line numbers, the source code is completely compatible between the two machines. (That's because both assemblers have the same origin.)

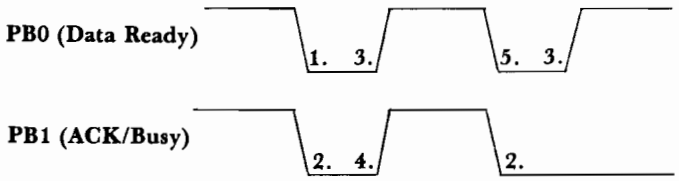
The software I'm presenting is a version which dumps object code from either the AIM, SYM or 6522 equipped APPLE to my KIM.

I'm not providing the source file transfer program because I've still got some bugs in it. (Maybe I'll print that routine some other time.)

One of the fastest and, perhaps, even the simplest method of transferring data from one computer to another is to do it in parallel. Each computer needs an 8-bit I/O port and several "handshaking" lines for signaling "data sent" and "data received". All of my systems have a user accessible I/O port (I recently installed a 6522 VIA in my Apple II) so all that I needed to do was hook up the lines and write the software. (It always turns out to be "easier said than done", however.)

The first problem turned out to be figuring out the proper "handshaking" sequence. I first looked at the popular "Centronics" style handshaking sequence but decided to simplify it down to two lines (instead of three).

Handshaking Sequence



XMTR starts first

1. **XMTR** initializes 'Data Ready' low and waits for the RCVR line 'Acknowledge/Busy' to go low.
1. **RCVR** initializes 'ACK/Busy' low and waits for the 'Data Ready' line to go high indicating that there is a BYTE available on the lines.
3. **XMTR** puts a data BYTE on the lines, sets the 'Data Ready' line high and waits for the RCVR 'ACK/Busy' line to go high signifying that the data has been received.
4. **RCVR** accepts a data BYTE and sets the 'ACK/Busy' high
5. **XMTR** sets 'Data Ready' low after 'Ack/Busy' goes high

If I had to do it all over, I would have added a third line to indicate that the byte on the lines was the last byte to be transferred. This would be better for transferring binary dumps since, in that mode, with only two handshake lines, the receiver has no way of knowing when the data transfer in completed and must be RESET to get it out of an infinite loop.

The neat handshaking modes available in the 6522 on the AIM weren't used because I wanted to be able to use the same software for both the KIM and the AIM and those special I/O operating modes aren't available on KIM since it uses a 6530 for its user I/O. (Although the example software is only used to send data one way-- from AIM to KIM, it has been used to send data the other way also).

As far as the hardware connection goes--simply hook PA0-PA7 on the KIM to PA0-PA7 on the AIM (PA0 to PA0, PA1 to PA1 etc), PB0-PB1 on the KIM to PB0-PB1 on the AIM, and then tie the system grounds together. That's not too difficult, is it?

IMPORTANT NOTE: *Both systems must be reset* to put the I/O lines in a known state (all lines go "high" after a system reset). The order in which the

programs are started is also important. The transmit program must be started first, then the receive program.

```
HDE ASSEMBLER REV 2.2

 LINE#   ADDR   OBJECT      LABEL  SOURCE            PAGE 0001
01-0010  2000               ;THIS PROGRAM TRANSFERS OBJECT CODE
01-0020  2000               ;OVER THE PARALLEL INTERFACE. THE ADDRESS
01-0025  2000               ;LIMITS OF THE DUMP MUST BE SETUP BY
01-0026  2000               ;THE USER IN POINT1 (START) AND
01-0027  2000               ;AND POINT2 (END+1).
01-0028  2000
01-0030  2000               ;WRITTEN BY ERIC C. REHNKE 9/80
01-0040  2000
01-0050  2000                      *=$0000
01-0055  0000               ;WORKING POINTERS
01-0056  0000'
01-0057  0000              POINT1 *=*+2
01-0060  0002              POINT2 *=*+2
01-0080  0004
01-0095  0004
01-0100  0004               ;6522 LOCATION
01-0105  0004
01-0110  0004              IOBASE =$A000
01-0120  0004              PBD    =IOBASE
01-0130  0004              PBDD   =IOBASE+2
01-0140  0004              PADD   =IOBASE+3
01-0150  0004              PAD    =IOBASE+15
01-0160  0004
01-0190  0004
01-0200  0004                      *=$200
01-0210  0200                      .OFF C000
01-0220  0200
01-0230  0200  D8                  CLD               ;DON'T EVER FORGET THIS!!!!!!!
01-0290  0201
01-0300  0201  A9 FF       INITTX  LDA #$FF          ;MAKE THE 'A' SIDE
01-0310  0203  8D 03 A0            STA PADD          ;ALL OUTPUTS
01-0320  0206  A0 00               LDY #0            ;CLEAR THE OFFSET
01-0330  0208  A9 01               LDA #1            ;SET PB0=OUTPUT (DATA READY)
01-0340  020A  8D 02 A0            STA PBDD
01-0350  020D  8C 00 A0            STY PBD           ;...AND MAKE IT LOW
01-0355  0210
01-0360  0210  AD 00 A0    CKLOOP  LDA PBD           ;WAIT HERE FOR THE RCVR
01-0361  0213  29 02               AND #2            ;TO BRING THE ACK/BUSY LOW AND
01-0365  0215  D0 F9               BNE CKLOOP        ;SIGNIFY THATS ITS READY.
01-0394  0217
01-0395  0217  A0 00       REENT1  LDY #0
01-0400  0219  B1 00               LDA (POINT1),Y    ;NOW GET A CHARACTER
01-0410  021B
01-0420  021B  20 2E 02            JSR XMTR          ;...AND SEND IT ACROSS.
01-0500  021E
01-0510  021E  20 4E 02            JSR INCPTR
01-0520  0221  A5 00               LDA POINT1        ;SEE IF WERE FINISHED
01-0530  0223  C5 02               CMP POINT2        ;BY COMPARING POINTERS
01-0540  0225  D0 F0               BNE REENT1
01-0550  0227  A5 01               LDA POINT1+1
01-0560  0229  C5 03               CMP POINT2+1
01-0565  022B  D0 EA               BNE REENT1
01-0610  022D
01-0620  022D  00                  BRK               ;RETURN TO MON WHEN DONE
01-0630  022E
01-0640  022E               ;TRANSMITTER SUBROUTINE
01-0650  022E
01-0660  022E  48          XMTR    PHA               ;SAVE THE CHARACTER
```

```
01-0670  022F  48              PHA             ;TWICE
01-0680  0230  AD 00 A0  ACKLP1 LDA PBD         ;WAIT TIL 'ACK/BUSY' IS LOW
01-0690  0233  29 02            AND #2
01-0700  0235  D0 F9            BNE ACKLP1
01-0710  0237
01-0720  0237  68               PLA             ;RECOVER DATA
01-0730  0238  8D 0F A0         STA PAD
01-0740  023B  A9 01            LDA #1          ;RAISE 'DATA READY' HIGH
01-0750  023D  8D 00 A0         STA PBD
01-0760  0240
01-0770  0240  AD 00 A0  ACKLP2 LDA PBD         ;WAIT TIL 'ACK/BUSY' IS HIGH
01-0780  0243  29 02            AND #2
01-0790  0245  F0 F9            BEQ ACKLP2
01-0800  0247
01-0810  0247  A9 00            LDA #0          ;NOW DROP THE 'DATA READY' LINE
01-0820  0249  8D 00 A0         STA PBD
01-0830  024C  68               PLA             ;RECOVER CHAR FOR CR TEST
01-0840  024D  60               RTS
01-0850  024E
01-0860  024E           ;HERE WE INCREMENT POINT1
01-0870  024E
01-0880  024E  E6 00    INCPTR  INC POINT1
01-0890  0250  D0 02            BNE EXIT
01-0900  0252  E6 01            INC POINT1+1
01-0910  0254  60       EXIT    RTS
01-0920  0255
01-0940  0255
01-0950  0255
01-0975  0255                   .END
```

```
HDE ASSEMBLER REV 2.2

 LINE#   ADDR   OBJECT     LABEL  SOURCE          PAGE 0001

01-0010  2000           ;THIS PROGRAM RECEIVES OBJECT CODE FILES
01-0020  2000           ;OVER THE PARALLEL INTERFACE AND STORES
01-0030  2000           ;THE DATA STARTING AT THE LOCATION
01-0040  2000           ;INDICATED BY THE POINTER AT $0000.
01-0050  2000           ;THIS POINTER MUST BE INITIALIZED BY THE USER.
01-0055  2000
01-0060  2000           ;WRITTEN BY ERIC C. REHNKE 9/80
01-0070  2000
01-0080  2000                  *=$0000
01-0090  0000          POINT1  *=*+2
01-0100  0002
01-0110  0002           ;6530 LOCATION
01-0115  0002
01-0120  0002          IOBASE  =$1700
01-0130  0002          PBD     =IOBASE+2
01-0140  0002          PBDD    =IOBASE+3
01-0150  0002          PADD    =IOBASE+1
01-0160  0002          PAD     =IOBASE
01-0170  0002
01-0190  0002
01-0200  0002                  *=$2000
01-0210  2000
01-0220  2000
01-0230  2000
01-0240  2000
01-0250  2000
01-0251  2000  D8              CLD             ;DON'T EVER FORGET THIS!!!!!
01-0260  2001  A9 00    INITRX  LDA #0          ;MAKE THE 'A' SIDE ALL INPUTS
01-0270  2003  8D 01 17         STA PADD
01-0280  2006  A0 00            LDY #0          ;CLEAR THE OFFSET
01-0290  2008  A9 02            LDA #2
01-0300  200A  8D 03 17         STA PBDD        ;SET PB1=OUTPUT (ACK/BUSY)
```

```
01-0310  200D  8D 02 17          STA PBD           ;AND MAKE IT HIGH
01-0360  2010
01-0370  2010  20 4D 20   CONT   JSR INCPTR        ;BUMP THE POINTER
01-0380  2013  20 1B 20          JSR RCVR          ;GET A DATA BYTE
01-0390  2016  91 00             STA (POINT1),Y    ;STORE IT
01-0400  2018  4C 10 20          JMP CONT          ;KEEP LOOKING FOR DATA
01-0430  201B
01-0440  201B  A9 00      RCVR   LDA #0            ;DROP THE 'ACK/BUSY' LINE
01-0450  201D  8D 02 17          STA PBD
01-0460  2020
01-0470  2020  AD 02 17   DRLP1  LDA PBD           ;WAIT FOR 'DATA READY'
01-0480  2023  29 01             AND #1            ;TO GO HIGH
01-0490  2025  F0 F9             BEQ DRLP1
01-0500  2027  20 54 20          JSR DELAY
01-0510  202A  AD 02 17          LDA PBD
01-0520  202D  29 01             AND #1
01-0530  202F  F0 EF             BEQ DRLP1
01-0540  2031
01-0550  2031  AD 00 17          LDA PAD           ;GET DATA
01-0560  2034  48                PHA               ;SAVE IT
01-0570  2035
01-0580  2035  A9 02             LDA #2            ;SET 'ACK/BUSY' HIGH TO
01-0590  2037  8D 02 17          STA PBD           ;SIGNAL 'DATA RECEIVED'
01-0600  203A
01-0610  203A  AD 02 17   DRLP2  LDA PBD           ;NOW WAIT FOR 'DATA READY'
01-0620  203D  29 01             AND #1            ;TO GO LOW
01-0630  203F  D0 F9             BNE DRLP2
01-0631  2041  20 54 20          JSR DELAY
01-0632  2044  AD 02 17          LDA PBD           ;...AND THEN HIGH.
01-0633  2047  29 01             AND #1            ;THIS SAYS "DATA READY !"
01-0634  2049  D0 EF             BNE DRLP2
01-0640  204B  68                PLA               ;RECOVER DATA
01-0650  204C  60                RTS               ;AND RETURN
01-0660  204D
01-0670  204D
01-0680  204D  E6 00      INCPTR INC POINT1
01-0690  204F  D0 02             BNE EXIT
01-0700  2051  E6 01             INC POINT1+1
01-0710  2053  60         EXIT   RTS
01-0720  2054
01-0750  2054             ;THIS IS A DUMMY DELAY ROUTINE
01-0760  2054             ;THAT WAS USED FOR TESTING PURPOSES.
01-0770  2054
01-0771  2054  60         DELAY  RTS
01-0775  2055
01-0780  2055                    .END
```

Multi-Computer/Multi-User Games

No, I'm not a computer game freak. But, I am excited about the fantasy role playing games that are becoming available for computers. The intriquing Dungeons and Dragons game really grabbed my interest. Almost from the time I first become aware of it, I was toying with ways to computerize certain aspects of it. Certainly, the dice throwing part could be computerized, as well as the bookkeeping aspects of the game--like keeping track of the character attributes and whether or not certain moves are legal as well as the relatively complicated procedure of deciding how much damage has been done by certain moves. Freeing the player from having to handle all the complex paperwork should make the game all that much more enjoyable. Any game freaks out there care to comment?

As I look around the field, I don't see too much being done in the area of multi-user/multi-computer games. Computer games have been in the man-against-computer mode for quite some time and have made computer hobbyists appear almost anti-social. It's time for a change.

A fellow at work and I are working out the details for a two-player/two-computer game which uses a couple of AIM 65 computers. The first game will be rather simplistic but it will serve to get things started. Anyone out there working along the same lines? Get in touch? Let's join fantasies.

I can picture a time when many computers are linked together playing a rather complex fantasy type game, or, perhaps a realistic simulation type game.

Software Review

How would you like to develop 1802 programs on your AIM 65? Or, how would you like to set up a library of MACROS which can be called from your assembly language programs?

If either, or both of these things interests you, then you'll be interested in a new software package for the AIM 65 called MACRO.

MACRO is actually a pre-processor that works in conjunction with the AIM 65 assembler. Its function is to accept a source file that contains macro calls, expand those macros by looking them up in a library file, and outputting a new source file with all the macros expanded so that the AIM 65 ROM assembler can assemble it.

The macro library file must be set up which defines all the macros which are to be used and must be memory resident at the time the input file is submitted for expantion: (makes AIM 65 sound like a large machine, doesn't it?)

Here's an example of what it looks like:

```
SAMPLE MACRO
INCD POINTER
SAMPLE MACRO DEFINITION
& INCD
INC!1
BNE* +4
INC !1 +1
&
SAMPLE MACRO OUTPUT
INC POINTR
BNE* +4
INC POINTR +1
```

(The '&' character is used both to start and terminate a macro definition)

Now that last little programming sequence (incrementing a double byte pointer) is something 6502 programmers do alot of.

The same technique can be used to set up a cross assembler for most any other CPU (6800, 1802, 8080 etc). Pretty excitin' stuff!!!

According to the documentation that accompanies MACRO, the minimum usable system is an AIM 65 with 2K of RAM, the assembler ROM, and remote control of least one cassette deck. The price is $15 which includes documentation and a cassette of the object code. The source code for MACRO is available either on cassette or as a listing (you must specify) for an additional $30. (This would enable you to adapt MACRO to your 6502 floppy system).

So far, I haven't found any bugs in the system (I'm good at finding bugs) and it worked right the first time I tried it.

It's available from: POLAR SOLUTIONS
Box 268
Kodiak, Al. 99615

"AID" From HDE

AID (Advanced Interactive Disassembler) is a disassembler in the truest sense of the word. AID takes a machine language program as input and creates an assembly language source file as output. (Just the opposite of an assembler).

The source file includes labels and even equates for externally referenced locations. The file can then be assembled like any other source file.

Think about it. Remember all the time you spent manually building an assembler source file from a machine language program?

I can sure remember wasting lots of time getting a conventional disassembly listing, writing in labels and then typing the whole thing into a text editor file just to be able to modify a piece of software.

Since AID lets the computer do this "dirty"work, the programmer is free to spend more time doing the work that needs a bit more intelligence.

The source files can be assembled with the assembler from HDE which is compatible with the MOS Technology Cross Assembler.

More information on this exciting new software product can be obtained from Hudson Digital Electronics, POB 120, Allamuchy, N.J. 07820. (201) 362-6574. AID costs $95 and works just great.

No, I haven't made a source file from Microsoft BASIC as of yet. But, I'm sure some of you have it in mind. ©

Caveat Interruptor or Placating a Rebellious KIM Without Sacrificing RAM

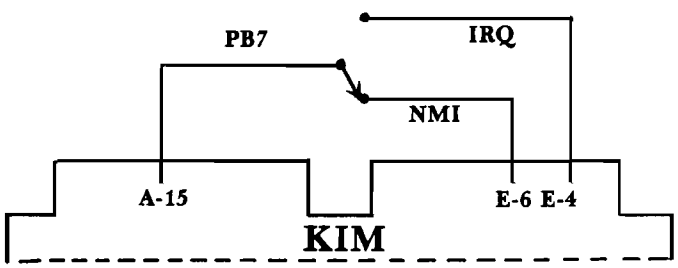
Philip K. Hooper

Summary: The dialog below presents a bizarre experiment that is easy to perform with a KIM-1, creating a runaway computer which no longer responds to 'reset'. Fortunately, the use of an unusual keypad sequence will restore normal monitor control. Moreover, KIM's peculiar behaviour is shown to be a perfectly proper response to an abnormal situation.

O: If you have your interval timer set up to generate a non-maskable interrupt, i.e. a connection between PB7 and NMI (Figure 1), then you can try a strange little experiment. Game?
I: Sure, I'll give it a try. What do I do?
O: Turn on the power and set your NMI vector (17FA,B) so it points to some very long program-like cassette read (1872) without input or the 4C4C4C loop at 4C4C. You can even try just leaving it as it 'comes up', and still the experiment will probably work.
I: That's nothing new. Sometimes I DO point NMI to 1873, so I can load in consecutive files by just pressing 'ST' between loads. And sometimes I don't even bother setting it. What next?
O: Take a look into location 0170.
I: So? Looks pretty unremarkable - just another meaningless byte of hex garbage.
O: That's right. Whatever your KIM happens to drop into that location at 'power up'. Now, pretend you are reaching for the 'AD' key but miss it and press 'C' instead.
I: Come on. I thought you said we were doing some sort of wierd experiement, but this is just stu . . . Hey! What's happening? The display is gone, and I can't seem to bring it back using the keypad. Even 'ST' doesn't help. But reset will always . . . What the devil IS this, anyway? How come all I get when I press the 'RS' key is a brief flash and then nothing? Have you tricked me into ruining my computer or something? What can I do? Isn't reset always supposed to bring KIM back in a known starting state?
O: It is - and it does. Of course, you could switch the power off and back on. That would probably put things right. But suppose you had just keyed in two full pages of code and were about to save them on tape when this pathological behaviour started. Surely you don't want to lose all that code and have to key it in again! Can you regain control without losing your RAM?
I: Well I don't see how. The only control I have is from the keypad, and not a single one of the keys does me any good!
O: True. Not a *single* key will help - but *three* will.
I: Eh?
O: Hold down 'ST', momentarily press 'RS' and then ' + ', and then release 'ST'.
I: Say that again.
O: No! You just go back and read it again, from two lines above this and then do it.
I: How about that. It worked! And without losing all that imaginary RAM. You know, this could have accidentally happened to me, and with you not around I would have had to turn it off to fix it. Say, what happened, anyway?
O: Well, when you pressed the 'wrong key', you inadvertantly addressed the interval timer, at 170C*, and it responded by generating an interrupt, i.e. a signal to follow the NMI vector 'somewhere'. Naturally, unless this 'somewhere' included a routine to sample and respond to the keypad, no keys other than 'ST' or 'RS' could possibly have had any effect. However, pressing 'ST' generates another NMI. Instead of helping get KIM back, it just sends it off to wherever it went before, again. 'RS' *does* bring it back, but only long enough to summon *another interrupt.* You see, when it returns control to the monitor, the monitor immediately accesses 170C again, unless the address stored in the pointer 00FA,B has been altered meanwhile.
However, holding 'ST' down will prevent recognition of this interrupt (the one invoked by the monitor after 'RS'), while pressing ' + ' will alter the pointer** so that the monitor no longer interrogates 170C. Then, since no further interrupt is being generated, releasing 'ST' restores normal operation at this point. Now, aren't you glad you asked?
I: Well, I . . . .
O: Say, give that ' + ' key another quick press, will ya;?

Author's Notes:
***several other addresses produce the same effect as 170C**
****the hex keys, '0' - 'F', and also 'PC', alter the pointer as well and may be used in place of the ' + ' key**

Figure 1 - Enabling the Timer Interrupt



Although a single wire between A-15 and E-6 is sufficient for the experiment explained above, a SPDT switch permitting the selection of either NMI or IRQ provides for more varied use of the timer interrupt.

COMMUNICATION!

Dann McCreary

While both the 1802 and the 6502 can handle quite a bit on their own, each has features which suit it to certain functions. Though the 1802 is not particularly fast, it has the advantages of low power consumption and low parts count needed to make a compact, portable system. On the other hand, the 6502 has the speed and software support for use as a powerful general purpose computer. Let's take a look at some ways to start a dialogue between an 1802 and a 6502.

Consider with me a few possible uses and layouts of COSMAC systems in communication with a central 6502 processor. One situation is the use of an 1802 to gather data from a remote location. The data would typically be transmitted to the main computer over a serial data link. This could take the form of a twisted wire pair, a radio transmission, a modulated light beam, telephone lines or even an intermediary like magnetic tape.

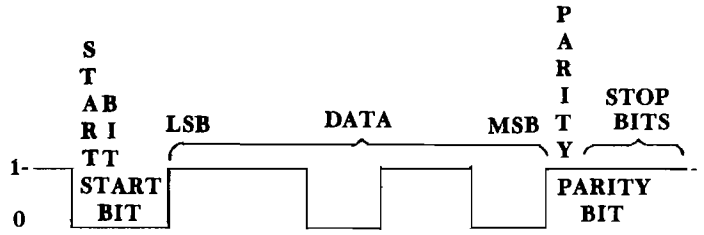
Another possibility is parallel communication. This would be used at closer range to achieve higher data rates. A parallel interface transfers an entire byte of data at a time. Some form of handshake is employed to coordinate the transfer timing. A portable 1802 unit might be brought and plugged into a central computer for a rapid transfer of data.

Perhaps the fastest and most direct communication between 6502 and 1802 could be obtained by combining the two processors as co-processors with common access to at least some memory regions. This would make possible the sharing of some tasks between the two processors. By setting or clearing specified bytes of shared memory, data might be passed from processor to processor and the activities of both coordinated.

Let's look at some serial data formats and the software considerations for producing them. The basic principal behind serial communication is to take a signal capable of presenting two states, 1 or 0, high or low, and to vary that signal in a specific time dependent pattern. This can be done readily by incorporating a UART such as the 1854 in your 1802 circuit. The 1854 is a CMOS UART (Universal Asynchronous Receiver / Transmitter). It has all the necessary circuitry on one chip for generating and interpreting serial data streams on a character by character basis. When connected to an 1802, the 1854 makes sending serial data as easy as outputting a byte of data to a selected port.

In the interests of keeping our 1802 system small and simple. However, let's do the following: we'll look at a way to use the Q line of our 1802 as a serial data output, and one of the External Flag lines as a serial data input. This eliminates the need for a UART, but it shifts the burden over to software.

What are the elements of serial data transmission that we must create by programming? Look at the illustration of an 8 bit data word in serial format. At the beginning of the word, the serial line is in a high (1) state. This high state is of an indefinite period of time. Transmission of the word is begun by bringing the line low for one bit-time. This is called the start bit. It is in effect saying, "Get ready guys-here comes the data!". The bit time is based on the desired data transfer rate, or "baudrate".

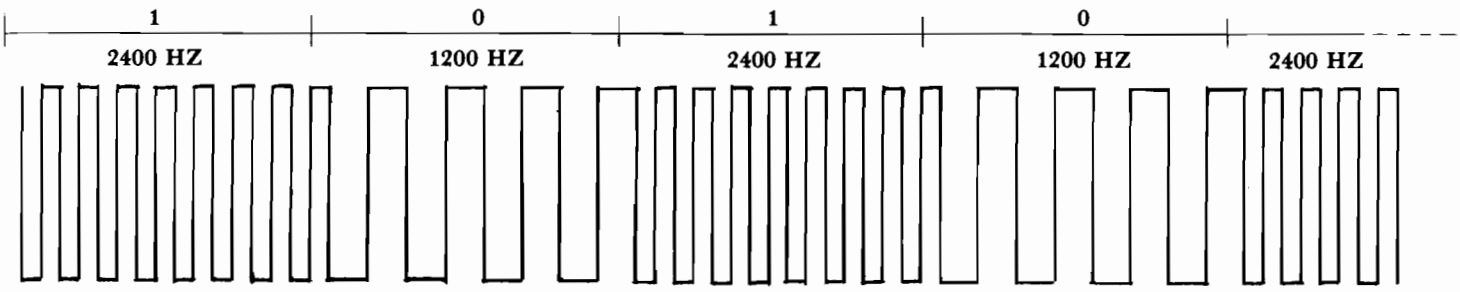


SERIAL REPRESENTATION OF 37_{16}

Following the start bit are 8 data bits, each using one bit-time. The first bit transmitted is the least significant bit of the data word. After the data bits comes a final parity bit. Finishing the transmission of the word are the stop bits. Stop bits are always 1 (high). For best reliability, 2 stop bits are recommended. This gives a receiver a fighting chance to synchronize itself with a continuous stream of data words. If a data word is not sent immediately, the line just remains high until the start bit of the next data word is sent.

For a variation on the theme, what if we wish to store the data on audio magnetic tape? We can use a very similar serial data format by superimposing audio tones onto our high and low segments of the signal. That is, let a high frequency tone represent a "1", and a low frequency tone a "0". The "Kansas City Standard" cassette format does in fact use this method. It differs from the above format only in that it does not use a parity bit. Each "0" consists of 4 cycles of 1200 HZ and each "1" consists of 8 cycles of 2400 HZ (see illustration).

Let's write a routine for generating either a straight serial data format or an audio-modulated cassette format. We'll set it up as a subroutine which, when called, will transmit the data in the "D" register in a serial format via the "Q" output flip-flop. We'll design our subroutine to allow for variation in the number of data bits. Parity will be settable as odd, even or completely off. The subroutine will also



PORTION OF A SERIAL BYTE IN AUDIO FORMAT

allow for either straightforward serial format or else audio-modulated serial format for use with a magnetic tape or telephone line transmission. In our next 1802 column, we'll examine some COSMAC code which will accomplish all this for us. ©

Book Review:"Son of Cheap Video" Author:Don Lancaster Publisher:Howard W. Sams, 1980 Price:$8.95

Reviewed by: Harvey B. Herman

To quote the author, "This is a you-build-it hardware book for hardware freaks...If you are not one of us, go away". I will assume that if you are still reading this review after seeing that quote that you will enjoy this book. It is intended for "poor folks" who like to tinker and construct useful things from a few chips and not much more. Specifically, it allows you to add a complete video display to a KIM-1 or the like for only $7 using five (count them) integrated circuits. Amazement is too mild a word for my reaction to that statement; flabbergasted is more like it.

The book is intended as a sequel to the author's earlier volume, "The Cheap Video Cookbook". Many references in the text to the earlier book suggest that it would be a good idea to have it close by to fully appreciate this effort. A legitimate criticism of the first circuitry concerned the amount of memory space used (28K bytes). What he now calls "scungy" video (I like the man's style) takes up 1K bytes for a 12x80 display - an impressive reduction in memory overhead.

A succession of projects is described in the book beyond scungy video. Lancaster shows how to combine cheap video with a "snuffler" coil on the outside of your TV set to free up processor time for normal comupting. This method locks the program and the display so picture jitter can be reduced with considerably less display program overhead. He includes a circuit for an EPROM programmer and describes how to use it in an extended music display example. Because the book leaves several projects as exercises (e.g. EPROM burning software) the book could be used as part of a course on microcomputers. Some of the construction hardware can be purchased from PAIA electronics (Oklahoma City, OK 73116) and could be conveniently provided to the students taking such a course.

I have not meant to leave the impression that the book is only for the KIM-1. Any of the enhanced-KIM clones (SYS or AIM) could benefit from the ideas in "Son of Cheap Video". Lancaster also includes chapters on 8080/Z80 systems, Heathkit H8, and Apple II (lower case display project). However, the book it not for every microcomputer owner as the initial quote suggested. Nevertheless it is well written, even entertaining in spots, can teach most of us a few things and save us money to boot. I recommend it highly. ©

Review:

Disk Operating System for KIM

Wilserv Industries $100
P.O. Box 115
Haddonfiled, NJ 08033

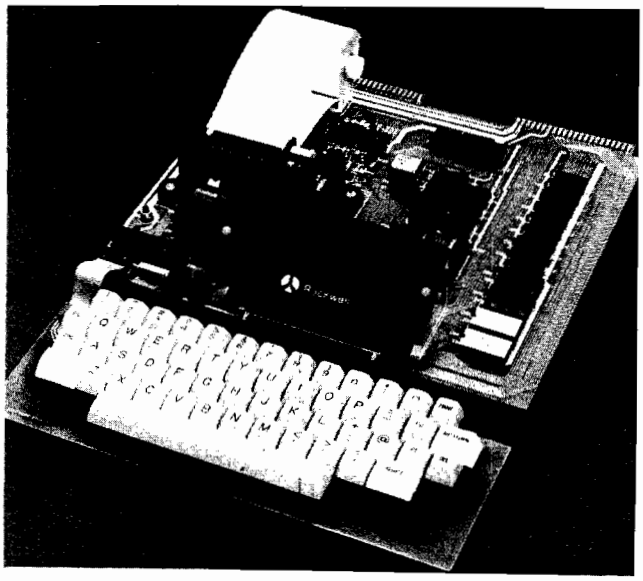
Reviewed by Harvey B. Herman

This is a short review of a disk operating system which has enhanced a KIM beyond my wildest dreams. I started with only a KIM - 1 but my system began to grow bigger and bigger almost immediately. Memory was added periodically, finally enough to use BASIC, using a KIMSI mother board. However, the weak link was the cassette operating system and the time it took to load programs. Switching to the Butterfield hypertape program helped but the delay (and occasionally bad loads) were irritating. I felt I really needed a better way to load and save programs. Wilserv had the answer.

Several years ago I purchased an Innovex 8'' disk and parts for a disk power supply. These sat around unused because an interface/controller to a minicomputer was never finished. To get the disk working on KIM, I needed a controller board (SDS Versafloppy I), a cable (made locally) and the software provided a Wilserv (Willi Kusche). To make a long story short, it works and I am very happy.

The KIM disk operating system is very convenient to use. It provides a link with BASIC and the same commands as the PET version. It provides a cheaper alternative for people like me who have most of the components already in hand. The only real disadvantages are the lack of random access files in the current version, and the element of do-it-yourself which does not appeal to everyone. Otherwise I recommend this software highly. ©

AIM 65 BY ROCKWELL INTERNATIONAL



AIM 65 is fully assembled, tested and warranted. With the addition of a low cost, readily available power supply, it's ready to start working for you.

AIM 65 features on-board thermal printer and alphanumeric display, and a terminal-style keyboard. It has an addressing capability up to 65K bytes, and comes with a user-dedicated 1K or 4K RAM. Two installed 4K ROMS hold a powerful Advanced Interface Monitor program, and three spare sockets are included to expand on-board ROM or PROM up to 20K bytes.

An Application Connector provides for attaching a TTY and one or two audio cassette recorders, and gives external access to the user-dedicated general purpose I/O lines.

Also included as standard are a comprehensive AIM 65 User's Manual, a handy pocket reference card, an R6500 Hardware Manual, an R6500 Programming Manual and an AIM 65 schematic.

AIM 65 is packaged on two compact modules. The circuit module is 12 inches wide and 10 inches long, the keyboard module is 12 inches wide and 4 inches long. They are connected by a detachable cable.

THERMAL PRINTER
Most desired feature on low-cost microcomputer systems . . .
- Wide 20-column printout
- Versatile 5 x 7 dot matrix format
- Complete 64-character ASCII alphanumeric format
- Fast 120 lines per minute
- Quite thermal operation
- Proven reliability

FULL-SIZE ALPHANUMERIC KEYBOARD
Provides compatibility with system terminals . . .
- Standard 54 key, terminal-style layout
- 26 alphabetic characters
- 10 numeric characters
- 22 special characters
- 9 control functions
- 3 user-defined functions

TRUE ALPHANUMERIC DISPLAY
Provides legible and lengthy display . . .
- 20 characters wide
- 16-segment characters
- High contrast monolithic characters
- Complete 64-character ASCII alphanumeric format

PROVEN R6500 MICROCOMPUTER SYSTEM DEVICES
Reliable, high performance NMOS technology . . .
- R6502 Central Processing Unit (CPU), operating at 1 MHz. Has 65K address capability, 13 addressing modes and true index capability. Simple but powerful 56 instructions.
- Read/Write Memory, using R2114 Static RAM devices. Available in 1K byte and 4K byte versions.
- 8K Monitor Program Memory, using R2332 Static ROM devices. Has sockets to accept additional 2332 ROM or 2532 PROM devices, to expand on-board Program memory up to 20K bytes.
- R6532 RAM-Input/Output-Timer (RIOT) combination device. Multipurpose circuit for AIM 65 Monitor functions.
- Two R6522 Versatile Interface Adapter (VIA) devices, which support AIM 65 and user functions. Each VIA has two parallel and one serial 8-bit, bidirectional I/O ports, two 2-bit peripheral handshake control lines and two fully-programmable 16-bit interval timer/event counters.

BUILT-IN EXPANSION CAPABILITY
- 44-Pin Application Connector for peripheral add-ons
- 44-Pin Expansion Connector has full system bus
- Both connectors are KIM-1 compatible

TTY AND AUDIO CASSETTE INTERFACES
Standard interface to low-cost peripherals . . .
- 20 ma. current loop TTY interface
- Interface for two audio cassette recorders
- Two audio cassette formats: ASCII KIM-1 compatible and binary, blocked file assembler compatible

ROM RESIDENT ADVANCED INTERACTIVE MONITOR
Advanced features found only on larger systems . . .
- Monitor-generated prompts
- Single keystroke commands
- Address independent data entry
- Debug aids
- Error messages
- Option and user interface linkage

ADVANCED INTERACTIVE MONITOR COMMANDS
- Major Function Entry
- Instruction Entry and Disassembly
- Display/Alter Registers and Memory
- Manipulate Breakpoints
- Control Instruction/Trace
- Control Peripheral Devices
- Call User-Defined Functions
- Comprehensive Text Editor

LOW COST PLUG-IN OPTIONS
- A65-010—4K Assembler—symbolic, two-pass **$79.00**
- A65-020—8K BASIC Interpreter **99.00**
- 3K RAM Expansion Kit **50.00**

POWER SUPPLY SPECIFICATIONS
- +5 VDC ± 5% regulated @ 2.0 amps (max)
- +24 VDC ± 15% unregulated @ 2.5 amps (peak) 0.5 amps average

PRICE: $389.00 (1K RAM)

Plus $4.00 UPS (shipped in U.S. must give ***street*** address), $10 parcel post to APO's, FPO's, Alaska, Hawaii. All international customers write for ordering information.

We manufacture a complete line of high quality expansion boards. Use reader service card to be added to our mailing list, or U.S. residents send $1.00 (International send $3.00 U.S.) for airmail delivery of our complete catalog.

RNB ENTERPRISES INCORPORATED

2951 W. Fairmount Avenue
Phoenix AZ. 85017
(602)265-7564

master charge VISA

Expanding KIM-Style 6502 Single Board Computers

Part 1 of 3:
Hal Chamberlin

Undoubtedly the most successful single board computer ever has been the KIM-1 made by MOS Technology (now Commodore). When introduced it apparently had just the right combination of features and price to attract tens of thousands of users. More recently of course the SYM-1 from Synertek and AIM-65 from Rockwell have incorporated numerous additional features into the same self-contained single board computer concept. Fortunately for users, all three of these machines are quite similar in their electrical characteristics.

Sooner or later however all computers need to be expanded and these single board machines are no exception. Although the SYM-1 and AIM-65 can be expanded somewhat merely by plugging in additional memory chips, the maximum limit is only 4K bytes of programmable memory. Thus additional boards are required for substantially increased RAM, ROM, or I/O capability. Recognizing this fact, the computer manufacturers as well as a number of independent accessory manufacturers have designed and brought to market a wide variety of expansion boards for the KIM, SYM, and AIM computers.

In most cases just having expansion boards available is not enough; there must also be a *motherboard* offered to plug them into since these computers have no on-board bus and slot sockets of their own. To date the computer manufacturers and independents have selected no fewer than four distinctly different ways to do this. First on the scene of course was MOS Technology who offered the KIM-4 expansion motherboard which mated with their KIM-2 and KIM-3 expansion memory boards. The bus presented by the KIM-4, which is called the "KIM-4 Bus", is in many ways similar to the bus presented by the computer itself as its own expansion edge connector. The primary difference is an altered pin assignment which is basically a one pin shift from the expansion connector assignments. This apparently was done to provide additional ground connections. Since then, independent manufacturers have also offered KIM-4 style expansion motherboards although there are important differences from the original KIM-4 (see Compute issue #3).

Shortly thereafter, as soon as the KIM's popularity became known, other independent manufacturers offered expansion motherboards which presented an S-100 style bus to the expansion boards. The primary advantage of this approach is that the user is not restricted to using expansion boards designed specifically for KIM-style machines but instead can choose from hundreds of S-100 compatible boards designed for 8080 based systems. Unfortunately many of the more sophisticated S-100 boards such as large dynamic memories, graphic display interfaces, and disk controllers could not be used because of substantial timing differences between 6502 and 8080 style microprocessors.

Late in 1977 Micro Technology Unlimited introduced a motherboard and card cage for the 6502 based single-board computers. The motherboard is little more than 5 edge connectors wired in parallel with one for the computer and the other 4 for expansion boards. The bus presented is the same pinout as that of the processor's expansion connector. The main advantage of this technique is the low cost and compact packaging afforded by the elimination of bus buffers. In addition, expansion boards compatible with this bus may be easily connected directly in parallel with the expansion connector if for some reason the motherboard is not desired. The main disadvantage is that the number of expansion boards is limited to four by the small drive capability of the computer's own bus.

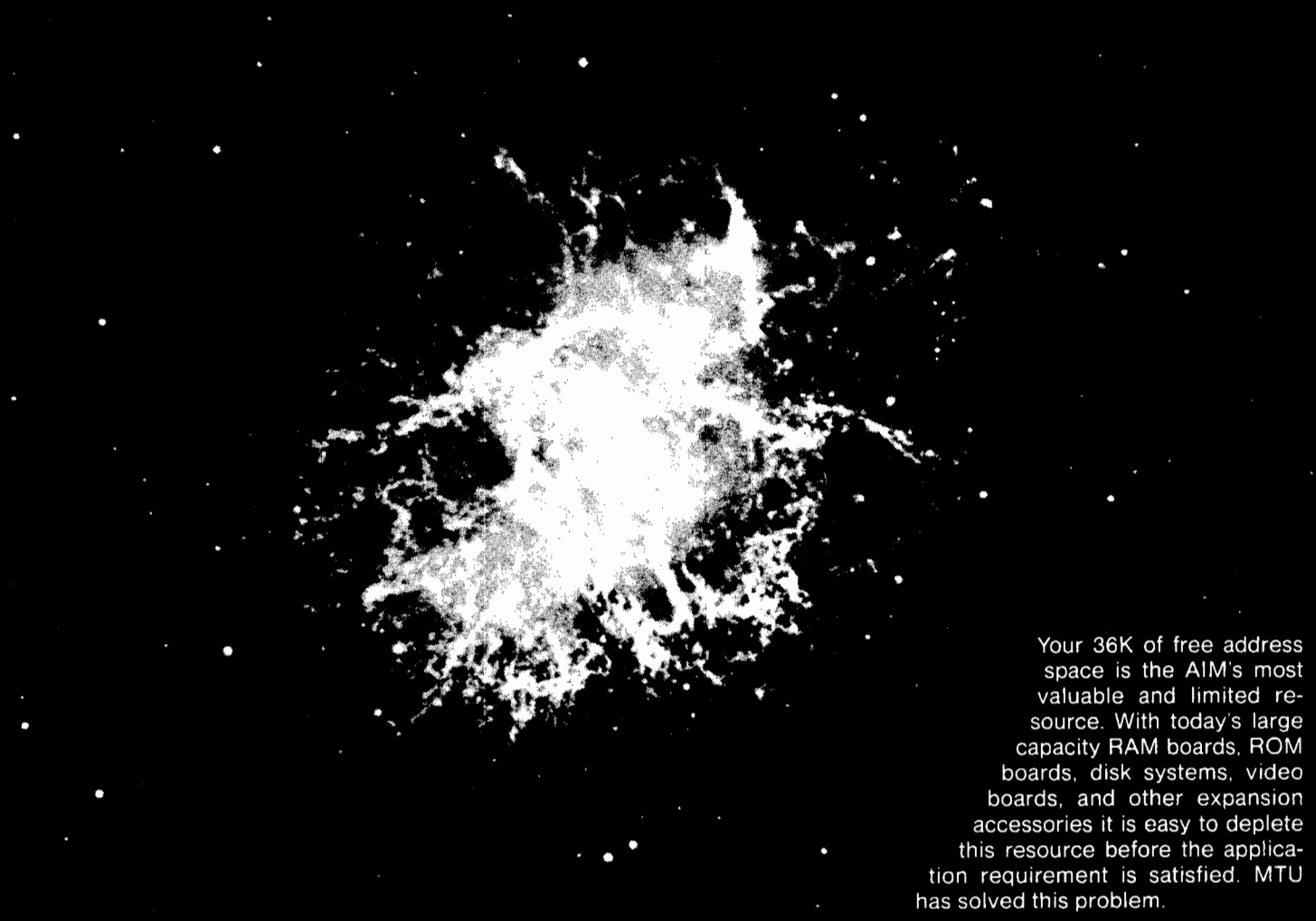
Recently Rockwell has introduced its expansion motherboard which essentially presents an Exorcisor bus to the expansion boards. Motorola originated this bus for use in their Exorcisor microprocessor development systems. Rockwell also uses the Exorcisor bus in their system 65 development system. The advantage of this method, at least to Rockwell, is avoiding the need to develop new expansion boards just for the AIM-65. To users the biggest drawback of the Exorcisor bus probably is the lack of reasonably priced boards to plug into it.

All four of these techniques are quite viable methods for expanding KIM-1, SYM-1, and AIM-65 single board computers and each has a broad base of dedicated users.

Mechanics

All three of the single board computers are intended to simply rest flat on a tabletop using the several quarter-inch high rubber feet provided. Although not the most beautiful thing in the world, it works well in many cases and is certainly inexpensive. In situations where better appearance is desired or small children are present, there are vacuum-forced dress covers available that simply slip over the computer board hiding everything except the display and keyboard.

A BRILLIANT FUTURE FOR YOUR AIM-65 WITH THE BANKER MEMORY ™



Your 36K of free address space is the AIM's most valuable and limited resource. With today's large capacity RAM boards, ROM boards, disk systems, video boards, and other expansion accessories it is easy to deplete this resource before the application requirement is satisfied. MTU has solved this problem.

THE BANKER MEMORY contains 32K of RAM, 4 PROM sockets for 2716/2732/2332, a PROM programmer, 40 bits of parallel I/O, and 4 timers from two 6522 I/O chips. Addressing is extremely flexible with the RAM independently addressable in 4K blocks, PROM's independently addressable, and I/O addressable anywhere on a 64 byte boundary (even in AIM's I/O area at AXXX by adding a single jumper to the AIM).

This may sound familiar, but read on! Unlike other AIM compatible memory boards, THE BANKER MEMORY has on-board bank-switching logic! The four 8K blocks of RAM plus the 4 PROM sockets make up 8 **resources**, each associated with a bit in an Enable Register. Through this Enable Register resources may be turned on and off under software control. When a resource is off, its address space is freed for other uses. You can even put BANKER resources at the same address and switch among them for virtually unlimited RAM and PROM expansion! You can even have multiple page zero's and stacks! Do you need 160K byte of memory? It only takes 5 of THE BANKER MEMORY boards and you end up with 5 page zeros and stacks to boot!

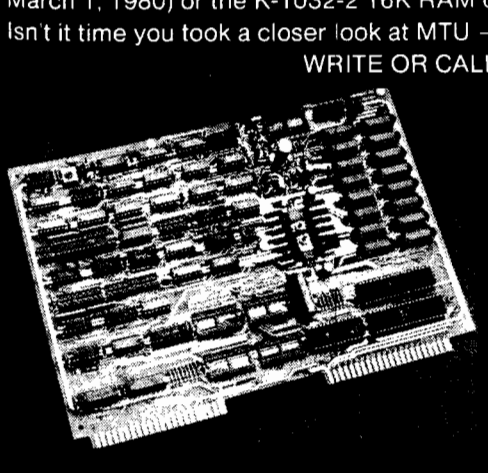
There's more! The BANKER MEMORY also incorporates 18 bit addressing which allows for the 256K address spaces of the future. RAM, PROM, and I/O each has its own full 18 bit address decoder which allows these resources to be in different 64K banks. This board and other MTU products, such as our 320 by 200 dot VISIBLE MEMORY and Floppy Disk Controller with 16K DMA RAM, can turn your AIM into a truly powerful 6502 computer that far surpasses the packaged systems in functional performance.

INTRODUCTORY SPECIAL K-1032-1 32K BANKER MEMORY FULLY ASSEMBLED AND TESTED $395.00 ($450.00 as of March 1, 1980) or the K-1032-2 16K RAM only with bank switching and 18 bit address bus only $295.00

Isn't it time you took a closer look at MTU — we offer you power now with an eye to the future.

WRITE OR CALL TODAY FOR OUR 48 PAGE FALL 1980 6502 CATALOG
International requests include $1.00

VISA and MASTERCARD accepted



MTU
Micro Technology Unlimited
2806 Hillsborough Street
P.O. Box 12106
Raleigh, NC 27605, U.S.A.
(919) 833-1458

The KIM-4, S-100, and Exorcisor type expansion motherboards simply extend this board-on-the-table concept. Typically the expansion motherboard is roughly the same size as the computer board and plugs straight onto its expansion connector. On the motherboard are perhaps a dozen integrated circuits for address decoding, bus buffering, and voltage regulation. The majority of the space however is taken up by 4 to 8 edge connectors which form the "slots" of the expansion bus. When plugged into these slots, the expansion boards assume a vertical orientation.

A system expanded this way uses a large amount of additional table space, and in the case of th KIM, it is useful space to the left of the computer. The assembly of interconnected boards is also rather fragile and certainly not portable unless dismantled (most people would probably bolt the computer and motherboard to a sheet of plywood or plastic to avoid this). In particular a stray elbow can do considerable damage if a board is knocked out of its slot during operation. Unfortunately the available plastic dress covers do nothing to protect the added motherboard or expansion boards.

Another approach that has been slowly gaining acceptance is to place the expansion boards underneath and parallel to the computer board. Thus the expansion motherboard, which ties all of the boards together, is vertical. In order to hold this assembly of boards together, an aluminum frame with card guides is typically supplied and the motherboard is attached to an opening in the frame. Figures 1 and 2 show the KIM and AIM versions respectively of Micro Technology's implementation of this concept.

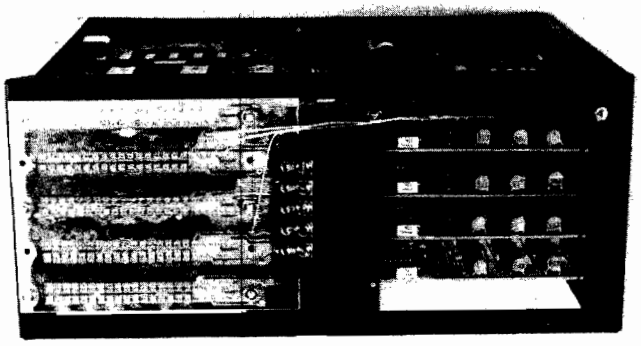


FIG. 1 KIM-1 INSTALLED IN A MICRO TECHNOLOGY UNLIMITED MOTHERBOARD/CARD FILE

The advantages of this configuration of course are reduced table space requirements and greatly increased protection for the expansion boards. The entire assembly of computer and boards is now one portable unit with only the power supply left over to worry about. The computer board is still exposed however. Probably the only potential disadvantage is that the computer's keyboard has been raised about 4 inches above the tabletop.

Electronics

There are electronic factors to consider as well when expanding a KIM, SYM, or AIM computer. In order to minimize cost, complexity, and power consumption, all three of these single board computers are designed without buffers between the microprocessor chip and the expansion edge connector. The KIM-1 went one step further and omitted part of the address decoding circuitry as well. The lack of buffers means that the expansion bus presented by these computers has a DC drive capability of only one standard TTL load, or equivalently, 5 low power Shottky loads. The AC drive capability depends on the desired signal rise-time. For bus operation at 1mHz, a total of approximately 25 "connections" at 6pF each can be driven. A connection here is defined as a gate input, disabled tri-state output, or MOS input (which does not contribute to DC loading).

Compared to other bus-oriented computers, such as S-100 machines, this does not sound like much of a bus at all since these machines typically have a drive capability of 30 standard TTL loads (74 series) or nearly 150 low power Shottky (74LS series) or over 200 low power TTL (74L series) loads. In fact, the original advertising for the MITS Altair computer boasted an expansion capacity of "over 200 boards". While this may have seemed necessary when using MITS's 1K memory and single port I/O boards, 10 slots is ample for even the largest S-100 setup when using today's dense memory and peripheral interface boards.

Over the years, experience has shown that several factors other than sheer driver power limit the number of boards that may be connected to a bus.

FIG. 2. AIM-65 INSTALLED IN A MICRO TECHNOLOGY UNLIMITED MOTHERBOARD/CARD FILE

The most serious of these is crosstalk noise between the bus address/data lines and the various bus con-

trol lines. This noise arises when large numbers of address and data lines change state simultaneously, which is a common occurrence. The fast voltage risetimes (around 5NS with the popular 8T97 drivers) and 50MA or greater surges of current along each changing line couple electrostatically and magnetically to other lines in the bus and on the expansion boards themselves. Longer busses and more boards plugged in gradually increase the crosstalk until noise on the control lines causes false triggering of memory and I/O boards and thus system failure. So severe is this problem that early S-100 systems would fail to operate even before the 16 board capacity of a single cabinet was reached.

A related, but much less severe problem, is signal reflection from the ends of the bus lines, which after all, act like transmission lines. This effect becomes significant when the signal transmission time exceeds about 1/2 of the signal risetime. At 1.5NS per foot with a 5NS risetime, the bus would have to be two feet long before termination was required. The apparent success of bus terminators sold for S-100 systems is probably due to their reduction of signal swings (the logic 1 level is limited to 3 volts and floating bus lines are pulled to 3 volts) which in turn reduces crosstalk noise.

From the author's experience in designing a large, fast bus oriented system (specifically the A. B. Dick Magna SL four terminal full-page word processing computer), there are three ways to solve bus noise problems. One is to thoroughly shield the bus with a full-width ground plane, or ideally, a three-layer motherboard with data/address on one side, ground in the middle, and control signals on the other side. This solves noise coupling on the bus but not on the expansion boards which in turn must be carefully designed to minimize their own crosstalk. This technique was used in the Magna SL machine because of speed requirements.

Another technique is to use filters and delays on the control signals obtained from the bus in order to reject narrow noise pulses. This technique can be extended to deal with any kind of noise problem at the expense of system speed and is the one typically used with minicomputers such as DEC PDP-11's and Data General NOVA's.

The third technique attacks the source of the noise, namely fast risetimes and large current surges, by using a low power bus. With liesurely risetimes of 50 to 100NS and drive capabilities of less than 10MA, such a bus is virtually noise-free and quite fast enough for normal microprocessor operation. This technique, coupled with some attention to groundplane shielding, is most applicable to unbuffered KIM/SYM/AIM expansion busses.

The foregoing is not meant to imply that all of the buffered expansion motherboards available for the KIM, SYM, and AIM computers are racked with noise. In fact, their bus length and number of slots is generally small enough to keep noise at tolerable levels. The major point is that high power drivers and indefinitely expandable busses do have drawbacks of their own.

The real question at this point then is: How many expansion boards can the unbuffered microprocessor bus drive before becoming overloaded? The 6502 microprocessor is rated to drive slightly more than 1 standard TTL load (equivalent to five low power shottky loads) on its address and data busses while most of the RAM's and ROM's tied to the data bus can drive two standard TTL loads. The 6520, 6522, and 6530 I/O chips have the same drive capability as the microprocessor. Thus in general the answer is at least four boards provided that the expansion boards themselves buffer the bus such that only one low power shottky load (.36MA in the zero state) is presented to the bus by the board. Many boards on the market and particularly those designed for an unbuffered bus do this. Actually, any well designed board would be expected to buffer the bus in order to provide clean signals for the remainder of the board logic. The reason that only four boards can be driven instead of five is that some of the address lines are loaded by a low power Shottky decoder IC on the computer board itself.

Next time: The Great Experiment