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.)

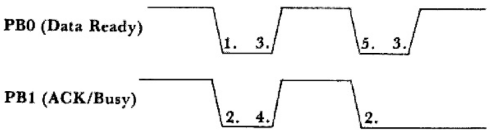
The software I'm presenting is a version which dumps object code from either the AIM, SYM or 6522 equipped APPLE to my KIM.

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

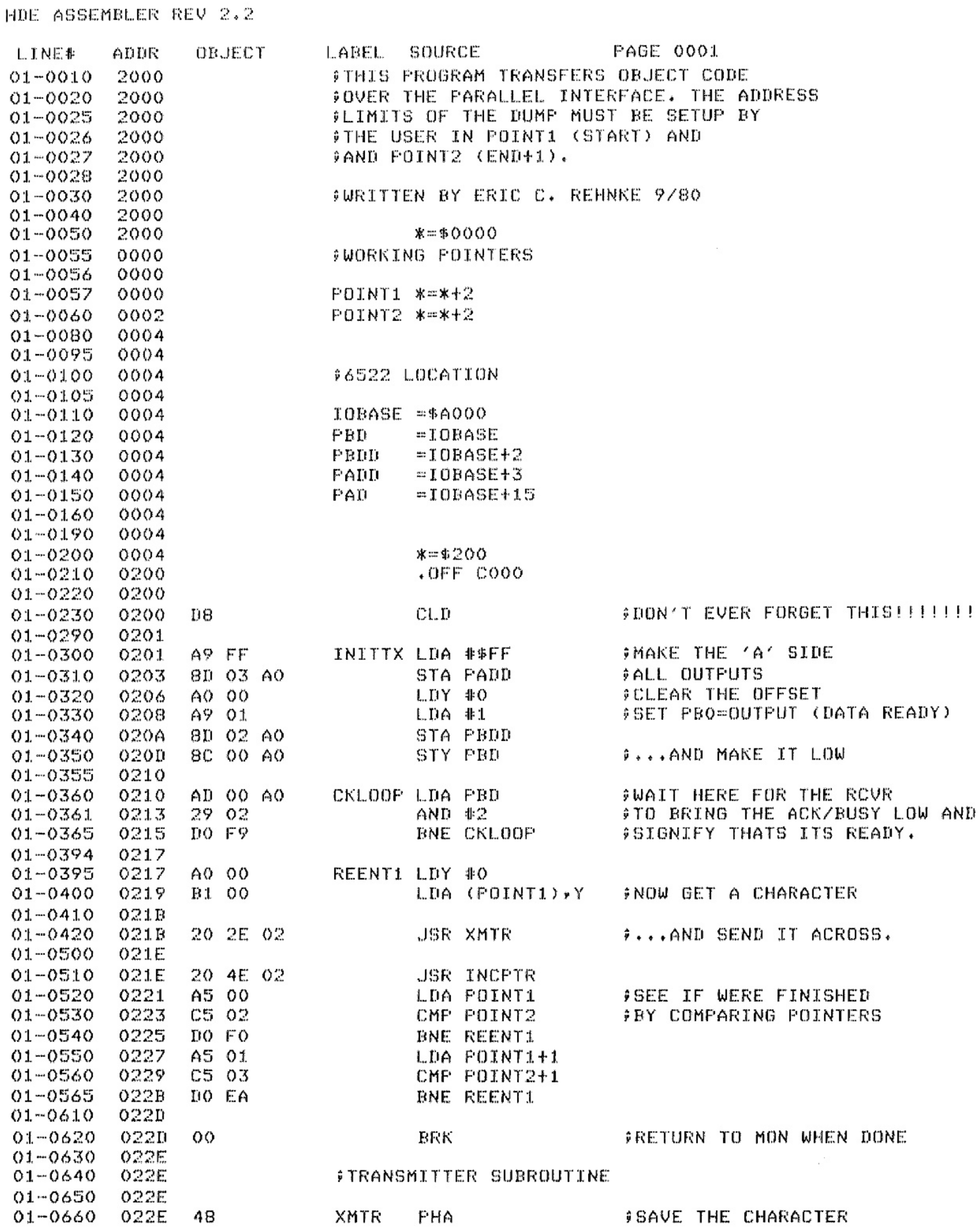
1. **XMTR** initializes 'Data Ready' low and waits for the RCVR line 'Acknowledge/Busy' to go low.
1. **RCVR** initializes 'ACK/Busy' low and waits for the 'Data Ready' line to go high indicating that there is a BYTE available on the lines.
3. **XMTR** puts a data BYTE on the lines, sets the 'Data Ready' 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. ©