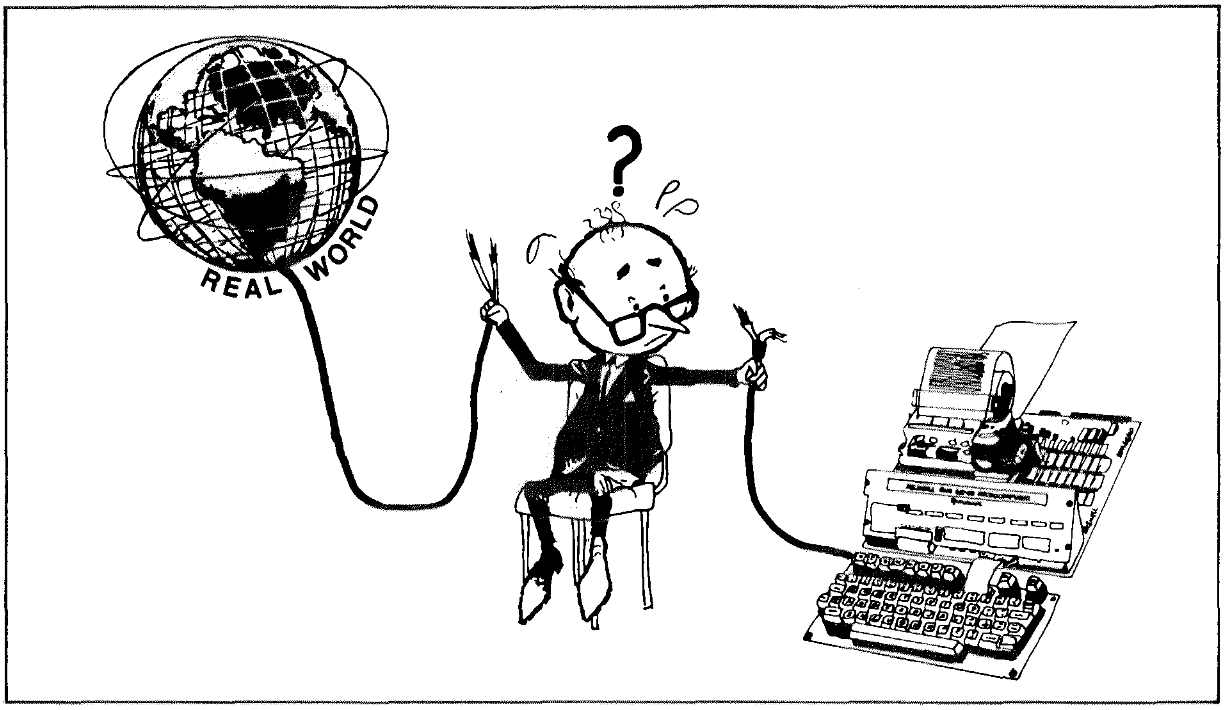
INTERACTIVE

APRIL, 1982

ISSUE NO. 8



If you are trying to interface your computer to the REAL WORLD, then you know how different that world is from our familiar digital world. We've included an article on page 3 of this issue as an aid in hooking up these two seemingly diverse worlds. We hope to offer more assistance in this area in future issues . . . with your help.

PLEASE NOTE: If you received this issue through the mail—you are on the INTERACTIVE mailing list and will continue to receive issues of this publication as they are issued.

Rockwell International

where science gets down to business

EDITOR'S CORNER

GOOD VIBES

I've received ALOT of favorable feedback concerning our special FORTH issue (#7).

One thing I've noticed about FORTH is that its users are as enthusiastic as early 6502 users. I partially credit this enthusiasm for the popularity the 6502 enjoys to this day. It would be great if FORTH were to become as popular in several years.

CHANGE OF ADDRESS

When you inform us of a change of address PLEASE give us your old address as well as your new address.

CORRECTIONS TO ISSUE #7

page 12—in the schematic a 1N4001 diode needs to be added so that the cathode is connected to pin 21 of the EPROM socket and the anode is connected to pin 24 of the EPROM socket. Also, a connection has to made from PB7 of the R6522 to pin 20 of the EPROM socket.

INSTANT PASCAL MANUAL IN GERMAN

I just received word that the AIM 65 Instant Pascal manual has been translated into the German language. It is available for DM30 from Hans-Joachim Regge, Fesenfeld 57, 2800 Bremen 1, West Germany (Telephone 0421-71114 abends)

MAIL ORDER

I just received a catalog from Mouser Electronics (11433 Woodside Ave., Santee, CA 92071) who lists all Rockwell's device and system level products. They say that they stock everything they list. Their phone number is 714-449-2222.

ERIC C. REHNKE—EDITOR

All subscription correspondence and articles should be sent to:

EDITOR, INTERACTIVE
ROCKWELL INTERNATIONAL
P.O. BOX "C"
4311 JAMBOREE RD.
NEWPORT BEACH, CA 92660

NEW LITERATURE AVAILABLE

The prices indicated for the following new literature is for cutotmers in the U.S. and Canada. All others need to add $2.00 per book ($1.00 per reference card or schematic) for shipping and handling. ALL PAYMENTS must be made in U.S. funds drawn on a U.S. bank (or by U.S. Postal Money Order). U.S. residents should add their appropriate state sales tax.

Order No.	Document	Price
283	AIM 65 FORTH Reference Card	$2.00
2100	AIM 65/40 8K Basic Reference Card	$2.00
289	AIM 65/40 System Summary Booklet	$2.00
280	AIM 65/40 System User's Manual	$15.00
297	AIM 65/40 Assembler User's Manual	$10.00
299	AIM 65/40 Basic User's Manual	$10.00
262	I/O ROM Program Listing (AIM 65/40)	$10.00
291	40 Character Display User's Manual (AIM 65/40)	$10.00
292	Graphics Printer User's Manual (AIM 65/40)	$10.00
288	Monitor/Editor Program Listing (AIM 65/40)	$10.00
290	AIM 65/40 SBC Module Schematic	$2.00
D78	AIM 65/40 Microcomputer Data Sheet	No Charge
RM23	AIM 65/40 Adapter Cable and Buffer Module Data Sheet	No Charge

COMPAS FOUND

After my little notice in the last issue about having trouble locating COMPAS MICROSYSTEMS, I received several phone calls and letters. Apparently, COMPAS has been purchased by the IMPACT SERVICES DIVISION of PIONEER HI-BRED INTERNATIONAL. According to a copy of an earlier letter I received, most of the system level products will be, or have been, phased out. For more information, the phone number of IMPACT SERVICES DIVISION is 515-270-3201

LETTERS TO THE EDITOR

Dear Editor,

I would like to thank Ken Fullbrook and Michael Chinn for their suggestions on the use of the Editor with BASIC. I have made good use of them. My INPUTs to BASIC consist of 2 and 3 fields, so I use special "flag" inputs when I want BASIC to shift gears. Also, the Editor will be restored to its top line if you PEEK(227) and 228 and POKE to 225 and 226 respectively.

C. M. Shaw
169 Rainbow Dr. NW
Ft. Walton Beach, FL 32548

(Continued on page 20)

REAL WORLD INTERFACE

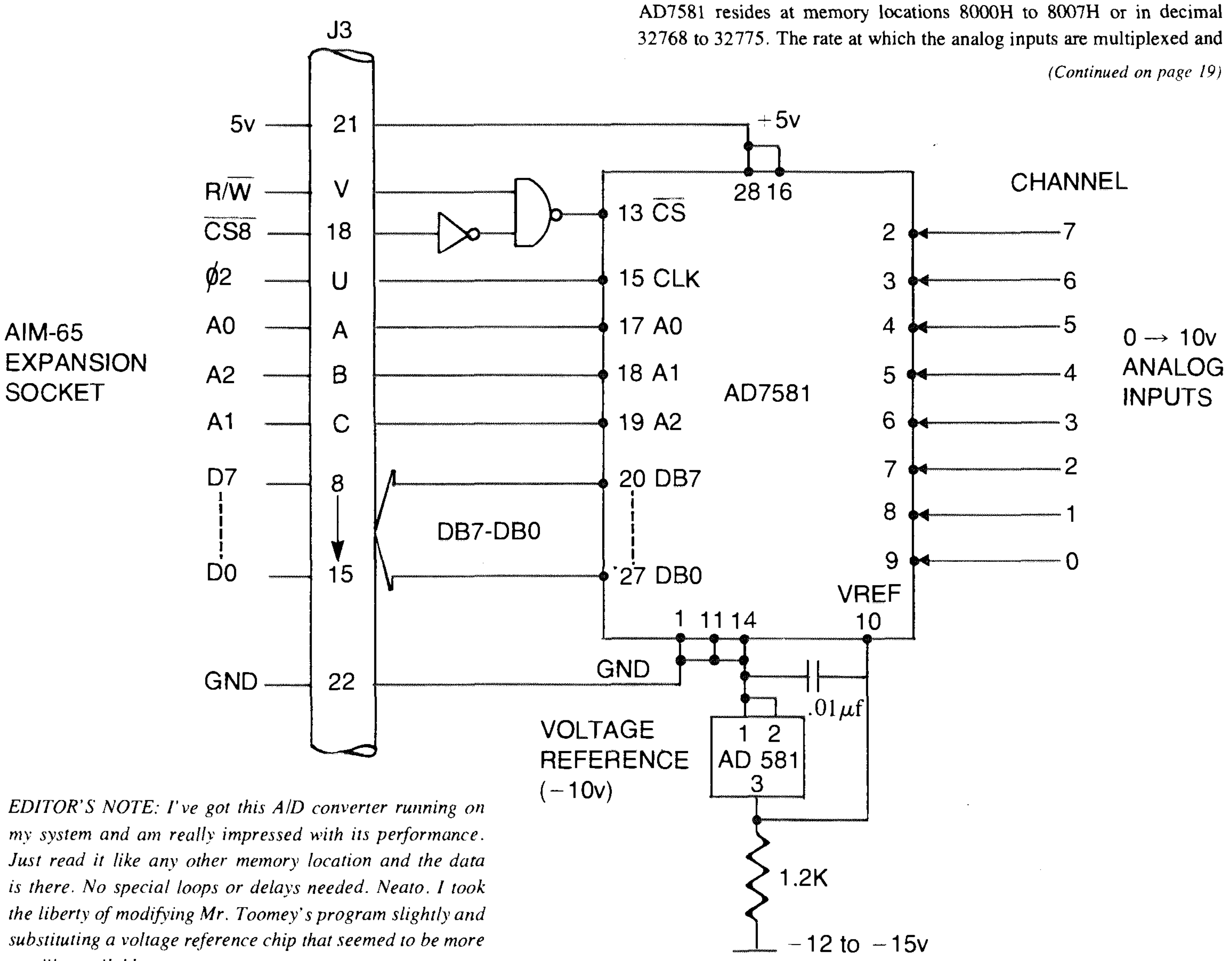
Paul Toomey
Applications Engr.
Analog Devices B.V.
Raheen Industrial Estate
Limerick, Ireland

A common task for microcomputers such as the AIM-65 is monitoring analog input signals from transducers. This brief note describes an easily constructed 8-channel analog input unit suitable for direct connection to the AIM-65 expansion port.

The heart of the analog input unit is a microprocessor compatible 8-channel, 8-bit CMOS data acquisition I.C. made by Analog Devices. The devices contains an 8-bit successive approximation A/D converter, an 8-channel multiplexer and an 8-bit dual port RAM. It continually scans and converts 8 analog inputs in a way that is totally transparent to the microprocessor. The user simply treats the device as 8 bytes of read only memory, each byte contains the conversion data for one analog input channel.

All of the control signals required are available at the AIM-65 J3 expansion port. The spare chip select $\overline{CS8}$ is gated with R/$\overline{W}$ to select the AD7581. Address lines A0, A1 and A2 address one of the 8 locations within the AD7581 corresponding to the 8 analog input channels. When selected the AD7581 puts the conversion data for the addressed channel onto the data bus by enabling its three-state data outputs. Thus the AD7581 resides at memory locations 8000H to 8007H or in decimal 32768 to 32775. The rate at which the analog inputs are multiplexed and

(Continued on page 19)



EDITOR'S NOTE: I've got this A/D converter running on my system and am really impressed with its performance. Just read it like any other memory location and the data is there. No special loops or delays needed. Neato. I took the liberty of modifying Mr. Toomey's program slightly and substituting a voltage reference chip that seemed to be more readily available.

AIM-65 8-CHANNEL ANALOG INPUT

SYSTEM SPOTLIGHT

AIM 65 BASED SYSTEM PROVIDES REAL TIME CONTROL WITH 32 I/O CHANNELS

Micro Interfaces Inc., Minneapolis, Minnesota, has an AIM 65 based Microcomputer Control System (MCS) and some associated hardware that might be of interest to INTERACTIVE readers.

MI's MCS has 16 input and 16 output channels, rated at 30 VDC at 3 amps. Each of the I/O channels are isolated from the AIM 65 signals and are noise suppressed.

Originally developed for behavioral research, the MCS's first application was controlling feeder and drinking devices, stimulus lights and recorder pens, while receiving inputs from lip-contact drinkometers and levers in Rhesus monkey cages.

However, the system can provide real time control, using BASIC language, for industrial processing, to turn on and off or pulse solenoids and lights, to record from DC switches, etc.

The MCS is made up of an AIM 65 microcomputer (A65-415, 4K RAM plus BASIC) plus three boards and a software ROM, housed in a single enclosure and weighing only twelve pounds.

The main addition is the MI-DCI interface board, with the 16 inputs and 16 outputs. An MI-J1 board provides audio cassette jacks, an RS232C or 20 ma TTY serial port, and internal connections to the DCI. An MI-J3 board allows for system expansion with the addition of memory boards, floppy disk drives and other devices.

An additional ROM allows real time programming of events and plugs into the assembler socket on the AIM 65. The software is an Interrupt Service Routine (ISR) which provides 10 millisecond resolution, an interrupt driven clock and numerous sub-routines that can be called by the user's BASIC program.

When coupled with all of the AIM 65 features—keyboard, printer, display, BASIC language, ROM, RAM, proven reliability—the MI additions make a very versatile control system. The system can even be expanded with the Rockwell RM series of microcomputer boards.

Micro Interfaces also offers a test box (MI-TB) with 16 outputs (LEDs) and 16 inputs (pushbutton switches) and interface cable with connector. This can be used to test programs and as a learning aid.

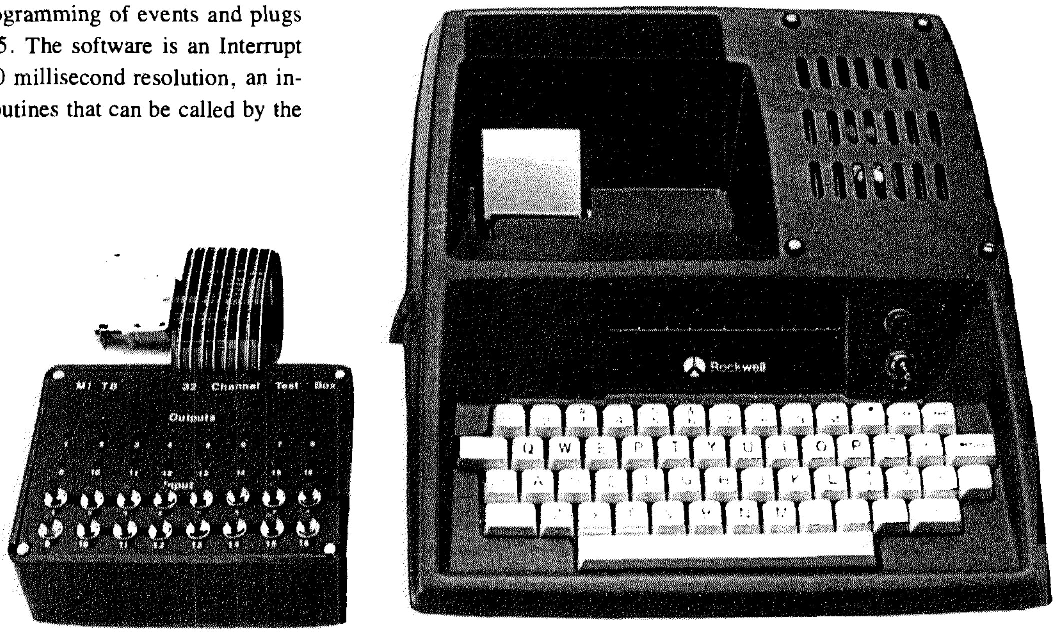
The boards MI designed to convert the AIM 65 into a control system may have some interest in their own right, particularly the MI-J1. This card connects to the J1 connector on the AIM 65 and provides two audio jacks, a 24-pin connector for the microcomputer's VIA signals, and an RS232C or 20 ma TTY serial port.

Current pricing has the MCS, including power supplies, the AIM 65, the three added boards and software ROM in an enclosure, at $1950. The MI-TB test box is $150. The MI-J1 board, fully assembled and tested is $29.95 (bare board is $13.95).

For more information, contact Micro Interfaces Inc., PO Box 14520, Minneapolis MN 55414. Or, call (612) 426-4603. We obtained our information from Peter Santi, MI's president.

Just to be sure, we want to add that Rockwell International has no connection with Micro Interfaces, and is not making claims or offering warranties for any MI products. We are trying to pass on to you information on AIM 65 applications that might prove of interest to INTERACTIVE readers.

Micro Interface's AIM 65 based Microcomputer Control System (MCS) in its enclosure, right, and the MI-TB test box, left. The MCS provides a 32 channel real time control system. The test box provides a matching 32 channels.



ERROR LOCATION IN FORTH

Stephen McHenry
16783 Beach Blvd.
Suite 303
Huntington Beach, CA 92647

As a development system, FORTH offers many advantages to AIM-65 users. However, some of FORTH's error messages are somewhat terse. This is especially true when an undefined reference is found in the program source. FORTH's response to this type of error is simply

```
    XYZ ?
```

with no additional assistance to help the user locate the error. For small programs, this is not usually a problem. However, when the program under development is reasonably large, locating the source of this error can be particularly frustrating since no source listing is generated giving the programmer the context of the word in error and no other clues are given.

However, if the source of the program is located in the memory of the AIM-65, one might reasonably expect that there should be a way of determining how to access the location where the error occurred. In addition, since AIM-65 FORTH uses the editor's line pointer as the position to start compilation, one might also reasonably expect that this pointer is also used to "remember" the "current" line during compilation. Thus, the line pointer would point to the vicinity of the error when compilation stops. This is, in fact, exactly what happens. Unfortunately, the only reentry command provided for the editor is "T", which automatically positions the line pointer to the top of the source program. What is needed is some way to access this information in a meaningful way before it is destroyed. There are two ways to do this.

Both methods involve adding a new word as the first word in the source program. This word is compiled, not to be referenced by any other words in the program, but rather to be invoked when compilation fails to aid in determining where the error occurred. When program development is complete, this word can be removed to make the program smaller.

The first word prints several lines of context surrounding the error and prints an arrow (<<--) at the end of the line containing the error. This is usually sufficient to locate the error in the source. The first line and last line printed may only be partial lines (the word could be changed to print only complete lines).

The actual source text for the word follows:

```
    HEX
    : WHERE
          CR
            00DF @
            30 -30 (SIZE OF CONTEXT)
            DO
                 I  -1 =
             IF   (Time to Print Arrow?)
                20 EMIT
                3C EMIT
                3C EMIT
                2D EMIT
                2D EMIT
            THEN
            DUP      (THE ADDRESS OF THE LINE POINTER)
            I  +     (ADD THE DISPLACEMENT)
            C@       (GET THE CHARACTER)
            EMIT
        LOOP
        DROP (THE ADDR OF THE LINE POINTER)
        CR ;
```

Once the context of the error is known, it is usually easy to locate the actual problem. The programmer can then exit from FORTH, enter the editor using the "T" command and quickly position to the point of error.

The word described above has proven itself useful in assisting programmers in quickly locating errors in large programs. However, some work is still required to leave FORTH, enter the editor and position to the error. It would be more desirable to actually do all of this with one word. The second word described actually exits FORTH and enters the editor at the point of the error (more specifically, at the line after the error).

The actual format of the word follows:

```
    HEX
    CODE EDIT
    EA24 JSR,
    F7B9 JMP,
    END-CODE
```

A clever observer will note that this is the same code that is executed when the "T" command is entered, minus the part which resets the editor's line pointer to the first line.

As soon as compilation halts due to an error, EDIT may be entered to return control to the editor. A "U" command should be entered to position to the line that actually caused the error. The error may then be corrected and the program recompiled.

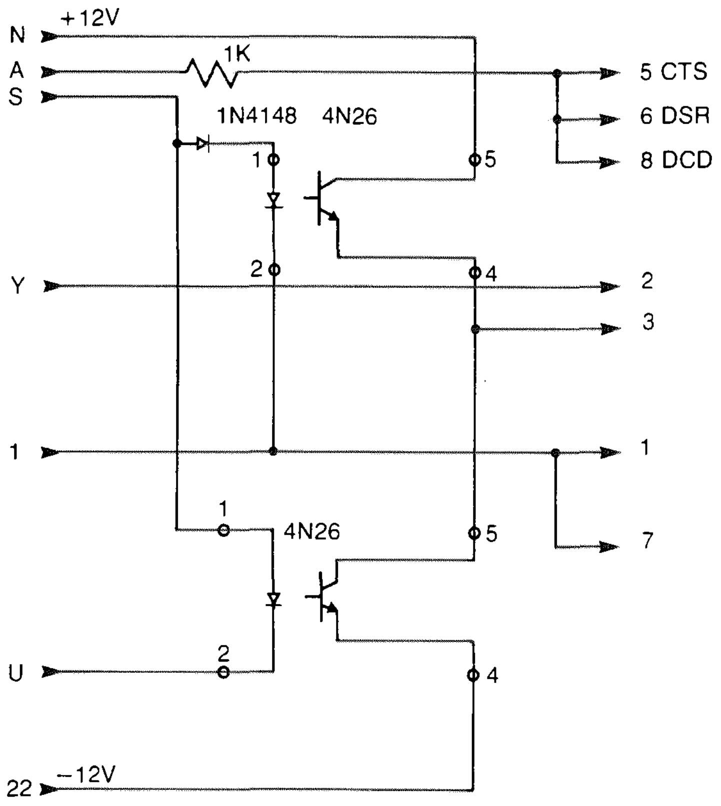
Of the two, the second word seems much more practical than the first. It is shorter, faster and more closely accomplishes exactly what is desired, entering the editor at the exact point of the error. However, both words have been included here for the sake of completeness and in the remote event that the second word cannot be used in some particular application.

MX80 & RS232

V.G.O.
Xerox Corp.

I found the program UOUT described in the issue No. 5 of INTERACTIVE to be very useful. However, I wanted to interface the EPSON MX-80 printer to AIM-65 and I had trouble with the printer being too slow to respond to a rather high stream of data sent to it by the AIM. The AIM must be able to detect when the printer is ready to accept data and to terminate the transmission when the printer is busy.

Here is how I solved this problem:



AIM-65 RS-232 INTERFACE

Instead of using another handshake line, the RXD line, pin 2 on the RS-232 connector could be used, since the printer does not have anything to send to AIM, anyway.

Pin 2 should be connected to the DTR line of the printer (11 and 20 on EPSON), and it will be tested for printer being READY. The UOUT program should be modified in such a way that DTR will be tested before the character is sent to the printer. If the printer is busy it will loop on the test. The addition to the program UOUT is shown on the enclosed listing.

If you tried the AIM RS-232 interface described in the Rockwell application note No. 230: RS-232 Interface for AIM 65, you might be surprised to discover that the implementation using the opto-coupler does not work for the baud rate higher than 1200. The same apply to the interface described in the issue No. 5 titled: EASY RS-232C.

The reason for this is that in the both bases a resistor is used as a pull-up for the transistor in the opto-coupler and the transition from −12V to +12V is slow.

If you still want to use the opto-coupler and would like to use the baud rate up to 9600 the remedy is shown on the figure.

I hope the readers will find some use for these items.

```
2000                OUTTY   =$EEA8
2000                CR      =$0D
2000                LF      =$0A
2000                NULL    =$00
2000                DRB     =$A800
2000                UOUT    =$10A
2000                        *=UOUT
010A  D0 2F                 .WORD START
010C                        *=$2FD0
2FD0                ;WAIT FOR DTR TO GO
2FD0                ; HIGH ON PIN 2
2FD0  90 19         START   BCC RETRN
2FD2  2C 00 A8      STUS    BIT DRB
2FD5  70 FB                 BVS STUS
2FD7  68                    PLA
2FD8  C9 0D                 CMP #CR
2FDA  D0 0C                 BNE NOTCR
2FDC  A9 0A                 LDA #LF
2FDE  20 A8 EE              JSR OUTTY
2FE1  A9 0D                 LDA #CR
2FE3  20 A8 EE              JSR OUTTY
2FE6  A9 00                 LDA #NULL
2FE8  20 A8 EE      NOTCR   JSR OUTTY
2FEB  60            RETRN   RTS
2FEC                        .END
```

CASSETTE RECORDER INTERFACING

Edward B. Hale
Assoc. Prof.
102 Physics Bldg.
University of Miss.
Rolla, Missouri 65401

The AIM 65 does not always interface well with cassette recorders, especially with El Cheapo (<$30) brands, but also with some more expensive models. For example, our El Cheapo recorder caused many error messages when the AIM was trying to read back the tape. Apparently, others are also experiencing problems as mentioned in INTERACTIVE, Issue No. 4. In solving our play back problem, several items of possible interest to readers of INTERACTIVE were discovered and, of greater importance, a simple circuit modification can substantially reduce data transfer errors and other recorder problems.

The input to the recorder (MIC) is approximately a square wave of frequency 1200 or 2400 Hz as described in Appendix F of the User's Guide. However, the output from the recorder (EAR) bears no resemblance to a square wave! This is basically because the recorder was not designed to transmit the many Fourier frequencies in a square wave. For example, our G.E. recorder transmits *only* the fundamental frequency of the 2400 Hz signal*. (The 7200 Hz and higher harmonic frequencies are all severely attenuated even when the treble is at its highest.) Unfortunately, our El Cheapo recorder had a better high frequency response and transmitted 7200 Hz. The transmission of these higher frequencies caused an undesirable ripple structure in the output signal. Occasionally, if this structure becomes noisy, it can be interpreted by the AIM circuitry as an extraneous (error) pulse. Unfortunately, no easy way was found to reshape the output signal to avoid this problem. However, a simple solution to the problem was found.

*This poor frequency response may be why the G.E. recorder works well.

The solution was based on reshaping the input signal from the AIM into a sinusoidal-like wave. The recorder handled without large distortion sinusoidal waves of both 1200 and 2400 Hz. These waves were easily converted by the AIM read-in circuitry back to square waves.

The input signals to the recorder was reshaped by integrating the signal using a capacitor. This integration actually produces a round-edge triangular wave, which is close enough to sinusoidal for this application.

more ⟶

The capacitor found to work best was 4.7 μf and was soldered directly to the edge connector between the AIM output (pin L) and ground (pin 1).

Installation of this capacitor has several advantages other than reducing false pulses. The circuit is no longer as sensitive to the volume control setting and works well over a much wider range of volume settings. Also, the Y/N test program in Section 9.1.4 of the User's Guide works well, which it did not do before.

It appears that this simple addition of an integrating capacitor will help solve many tape read back problems.

(EDITOR'S NOTE: This method solved the problem one AIM 65 user was having trying to read KIM-1 tapes.)

DILINK DILEMMA

C. M. Shaw
169 Rainbow Dr. NW
Ft. Walton Beach, FL 32548

When using the Dilink vector, $A406/A407, and AIM 65 goes off into never-never land, pressing RESET seldom, if ever, returns you to the Monitor unless you do the following:

Prior to changing the Dilink vector, change $A402 to something other than $7B.

Now, when you press RESET, line 382 of the Monitor program will detect a cold start condition and will reinitialize the Dilink vector to point to Outdis, $EF05.

Ideally, your program using the Dilink vector, should re-initialize $A402 (low byte of NMIV2) and the Dilink vector when re-started to avoid having to reload it.

If you plan to use the STEP/TRACE mode of operation, execute a RESET first to re-initialize Dilink and NMIV2.

INPUT/OUTPUT DEBUG ON THE AIM 65/40

Jim Thompson
Rockwell International

Trying to debug a program which transmits data to or from an external device is sometimes very difficult. This may be due to the effort required to access the data or the data may even be lost due to program malfunction. To aid in tracking this data transfer, the following programs were created to copy into memory each byte as it is transfered. The following description of operation is valid for either the input or output routines.

Locate the address of the input (output) vector of the device to be traced in the I/O vector table located at $0200.

For the device "S" $0204 ($0206)

Put this address into the instructions located at $71C ($740) and $721 ($745).

```
{M}071C    8D FE FF A9 07 8D FF FF  ..........
{/}071C       04 02          05 02
```

Using this address obtain the vector of the device JMP table.

```
{M}0204    62 F0 8B F0 56 F0 9D F0 b. . . V . . .
```

Now using this vector, display the device JMP table.

```
{M}F062    4C 6E F0 4C 63 F9 4C 6E  Ln.Lc.Ln
{  }F06A   F0 4C 5E F5 18 60 4C 17  .L . . 'L.
```

Copy the operand of the first JMP into the JMP to OPEN at $74A ($753).

```
{M}074A    4C FF FF 4C 00 07 4C FF L. . L. . L.
{/}074A       6E F0
```

Next copy the operand of the second JMP into the subroutine call at $700 ($726).

```
{M}0700    20 FF FF 8D FF FF 08 EE  ..........
{/}0700       63 F9
```

And finally, copy the operand of the third JMP into the JMP to CLOSE at $750 ($759).

```
{M}0750    4C FF FF 4C FF FF 4C 26 L. . L. . L&
{/}0750       6E F0
```

When you are ready to begin the copy process, execute the program located at $711 ($735). This will ask you for the starting address of the copy buffer and will link the trace program into the MONITOR. A COLD RESET is required to restore normal operation. Now get out there and remove all of those nasty I/O BUGS.

```
0726  8D FF FF   TRACOT  STA MEMORY
0729  EE 27 07           INC TRACOT+1     ; STEP TO NEXT CELL
072C  D0 03              BNE TRACO1
072E  EE 28 07           INC TRACOT+2
0731  28         TRACO1  PLP
0732  4C FF FF           JMP PUT          ; SEND BYTE TO OUTPUT DEVICE

0735  20 35 AE   SETOUT  JSR FROMX        ; GET ADDRESS OF COPY MEMORY
0738  8E 27 07           STX TRACOT+1     ; SET COPY ADDRESS
073B  8C 28 07           STY TRACOT+2
073E  A9 53              LDA #<TABOUT
0740  8D FE FF           STA OUTVEC
0743  A9 07              LDA #>TABOUT
0745  8D FF FF           STA OUTVEC+1
0748  00                 BRK
0749  EA                 NOP
```

```
AE35=               FROMX  = $AE35
FFFF=               GET    = $FFFF
FFFF=               PUT    = $FFFF
FFFF=               MEMORY = $FFFF
FFFE=               INVEC  = $FFFE
FFFE=               OUTVEC = $FFFE

0000                       *=$700
0700  20 FF FF      TRACIN JSR GET          ;GET A BYTE FROM THE INPUT DEVCE
0703  8D FF FF      TIN    STA MEMORY       ;..NOW COPY IT INTO MEMORY
0706  08                   PHP
0707  EE 04 07             INC TIN+1        ;STEP TO NEXT CELL
070A  D0 03                BNE TINXT
070C  EE 05 07             INC TIN+2
070F  28            TINXT  PLP
0710  60                   RTS

0711  20 35 AE      SETIN  JSR FROMX        ;GET ADDRESS OF COPY MEMORY
0714  8E 04 07             STX TIN+1        ;SET STORE MEMORY ADDRESS
0717  8C 05 07             STY TIN+2
071A  A9 4A                LDA #<TABIN      ;SET INPUT VECTOR TO COPY JMP
071C  8D FE FF             STA INVEC
071F  A9 07                LDA #>TABIN
0721  8D FF FF             STA INVEC+1
0724  00                   BRK
0725  EA                   NOP

FFFF=               OUTPUT = $FFFF
FFFF=               INPUT  = $FFFF
FFFF=               OPEN   = $FFFF
FFFF=               CLOSE  = $FFFF

074A  4C FF FF      TABIN  JMP OPEN
074D  4C 00 07             JMP TRACIN
0750  4C FF FF             JMP CLOSE


0753  4C FF FF      TABOUT JMP OPEN
0756  4C 26 07             JMP TRACOT
0759  4C FF FF             JMP CLOSE


075C                       .END
```

ADVENTURE!!!

Michael J. Thunquest
1067 East Austin Ave.
Salt Lake City, Utah 84106

(EDITOR'S NOTE: This is really a fun game for you lunch time adventurers! I was amazed at how challenging it was for so small a game. No, I haven't gotten through the whole game yet. For all you folks who don't look forward to typing the program in, Mike is making tapes available– Send him a S.A.S.E. for price information.)

Now the exciting adventure series comes to 4K Aim-65 in BASIC! This mini-version of those exciting games created by Scott Adams provides all the fascination and thrill of their "father" programs, but without their gigantic and confusing data files. This game is so action packed that there isn't even room in the program for directions! So here they are (very simple anyway).

At the beginning of the game you have been thrown, by forces beyond your control, into a dark underground cavern with tunnels leading to other rooms (5 total). These rooms contain odd pieces of equipment which will definitely come in very handy to you. The object of the game is to escape from the caves (using the tools you discover) carrying the precious treasures hidden there. The fewer the moves it takes you to accomplish this, the closer to a perfect score you will attain.

Use the commands E, W, N, S as directions for moving from cave to cave, otherwise use 2-word commands (a verb and a noun) to accomplish your tasks. If you type 'look' you will be shown again the room in which you reside.

If you are typing the program in by hand be sure to type it exactly as shown (including trailing spaces) to assure proper printout when playing. As soon as you have typed 'RUN' the game has begun. Try different commands as though you were really in that situation. You'll get the hang of it quite fast even if you've never been exposed to the Adventure games. Good Luck!

```
1 DIMO(12):O(2)=1:O(3)=1:O(5)=2:O(6)=2:O(1)=2:O(4)=3
3 DATA"SEE",1,"BOT",2,"WAN",3,"ROC",4,"LAN",5,"WAT",6,"GOL",7
4 O(10)=5:R=3:GOSUB37
5 DATA"SIL",9,"SAN",10,"DIA",11,"VEN",15,"FIS",16
6 DATA"PLA",18,"STE",19
8 DATA"E",1,"W",2,"N",3,"S",4,"FOL",12,"POU",6
9 DATA"CRO",12,"CLO",11
10 DATA"GET",5,"DRO",6,"THR",6,"FIL",13,"EMP",6,"PLA",6,"WAV",8
11 DATA"WAT",14,"CLI",9,"LOO",10,"PLU",11,"FIN",12,"JUM",12
12 S=0:F=0:B$="":C$="":M=M+1:INPUTA$
14 FORX=1TOLEN(A$):IFMID$(A$,X,1)<>" "THEN17
16 B$=LEFT$(A$,X-1):C$=RIGHT$(A$,LEN(A$)-X)
17 NEXTX:IFB$=""THENB$=A$:C$=""
19 Y$=LEFT$(B$,3):Z$=LEFT$(C$,3)
20 FORX=1TO14:READD$,D:IFD$=Z$THENS=D
22 NEXTX
24 FORX=1TO21:READD$,D:IFD$=Y$THENF=D:RESTORE:GOTO30
25 NEXTX:PRINT"YOU CANT ";A$:RESTORE:GOTO12
28 PRINT"YOU CANT ";A$:RESTORE:GOTO12
30 IFF>7THENF=F-7:GOTO34
32 ONFGOSUB77,83,87,92,100,135
33 GOTO12
34 ONFGOSUB220,250,265,290,275,280,285
36 GOTO12
37 ONRGOTO38,40,41,43,39
38 PRINT"THIS IS THE PIRTAES ROOM":GOTO49
39 PRINT"YOURE IN THE MYSTERYROOM":GOTO320
40 PRINT"YOU ARE AT A POOL OFWATER":GOTO49
41 PRINT"AN ESCAPE HOLE IS    HIGH ABOVE YOU":GOTO49
43 IFC(3)=0THEN46
45 PRINT"THIS IS A LARGE ROOM":GOTO49
46 PRINT"A VENT IS BLOWING   OUT STEAM. YOU CANT SEE ANY";
47 PRINT" OBJECTS IN  THE ROOM":RETURN
49 PRINT"THERE IS"
```

```
50 FORX=1TO12:IFO(X)<>RTHEN70
52 IFX>6THENY=X-6:GOTO54
53 ONXGOTO55,56,57,58,59,60
54 ONYGOTO61,62,63,64,65,66
55 PRINT"A SEED":GOTO68
56 PRINT"A BOTTLE":GOTO68
57 PRINT"A MAGIC WAND":GOTO68
58 PRINT"A ROCK":GOTO68
59 PRINT"A LIT LANTERN":GOTO68
60 PRINT"WATER":GOTO68
61 PRINT"GOLD!":GOTO68
63 PRINT"SILVER!":GOTO68
64 PRINT"SAND":GOTO68
65 PRINT"DIAMONDS!":GOTO68
66 PRINT"A HUGE PLANT"
68 L=1
70 NEXTX:IFL=0THENPRINT"NOTHING"
71 L=0:RETURN
72 IFR=3ORO(5)=6ORO(5)=RTHEN37
73 PRINT"ITS PITCH BLACK HERE"
74 PRINT"YOU BETTER GET BACK"
75 INPUTA$:IFRND(1)>.7THENR=2:O(5)=2:GOTO37
76 PRINT"TRY AGAIN":GOTO75
77 IFR<>1THEN79
78 R=2:GOTO72
79 IFR<>4THEN82
80 IFC(1)THENR=3:GOTO72
81 PRINT"A DEEP FISSURE BLOCKS THE WAY":RETURN
82 PRINT"WRONG WAY":RETURN
83 IFR=2THENR=1:GOTO72
84 IFR<>3THEN82
85 IFC(1)THENR=4:GOTO72
86 GOTO81
87 IFR=5THENR=1:GOTO72
88 IFR=3THENR=2:GOTO72
89 IFR<>4THEN82
90 IFC(2)THENPRINT"ROCKS HAVE BLOCKED  THE TUNNEL":RETURN
91 R=5:GOTO72
92 IFR<>2THEN95
93 IFO(10)=2THENR=3:GOTO72
94 PRINT"ITS TOO SLICK RIGHT NOW":RETURN
95 IFR=1THENR=5:GOTO72
96 IFR<>5THEN82
97 IFC(2)=0THENR=4:GOTO72
98 GOTO90
100 IFS>11THENPRINT"NO WAY!":RETURN
105 IFO(S)<>RTHENPRINT"I SEE NO ";C$:RETURN
110 IFS=6ANDO(2)<>6THENPRINT"YOU HAVE NO BOTTLE":RETURN
115 IFO>2THENPRINT"SORRY 3 IS THE LIMIT":RETURN
120 PRINT"OK":O(S)=6:O=O+1
125 IFO(7)=0ANDS=6THENPRINT"I SEE GOLD!":O(7)=R
130 RETURN
135 IFS>11THEN100
137 IFF=6ANDY$<>"DRO"THENS=6:C$="WATER"
```

(continued on next page)→

BOARD-LEVEL SEMINARS

Here is an up-to-date schedule of one-day seminars which will cover Rockwell AIM 65, AIM 65/40 and RM 65 product lines. Contact the appropriate person in your area for more information. The fee for the seminar will be $15 per person.

DATE	LOCATION	ROCKWELL REPRESENTATIVE CONTACT	TELEPHONE NUMBER
5/25/82	OTTAWA, CANADA	MARTY KENT	(416) 494-5445
5/26/82	MONTREAL, CANADA	MARTY KENT	(416) 494-5445
5/27/82	CALGARY, ALBERTA	BILL BISSONNETTE	(403) 483-6266
5/28/82	REGINA, SASKATCHEWAN	BILL BISSONNETTE	(403) 483-6266
6/1/82	SADDLEBROOK, NJ	PETE FERENTINOS	(516) 360-0940
6/2/82	COMMACK, NY	PETE FERENTINOS	(516) 360-0940
6/3/82	RICHMOND, VA	SHARON BULOVA	(703) 534-7200 or (800) 572-0405
6/8/82	CHICAGO, IL	BRAD HERRING	(312) 297-8862
6/9/82	MILWAUKEE, WI	TED CESSOP	(414) 257-2928
6/10/82	MINNEAPOLIS, MN	TW JOHNSON	(612) 544-7404
6/15/82	SEATTLE, WA	KRISTI SELLERS	(206) 454-9699
6/16/82	PORTLAND, OR	KRISTI SELLERS	(206) 454-9699
6/17/82	SUNNYVALE, CA	LARRY ADAMS	(408) 734-9865
6/22/82	HOUSTON, TX	CHUCK DAVIS	(713) 524-0528
6/23/82	DALLAS, TX	DEBORAH ULRICH	(214) 385-8885
6/24/82	PHOENIX, AZ	DON BULLOCK	(602) 971-6250
6/28/82	SAN DIEGO, CA	LOUISE TOUISSANT	(213) 478-0183
6/29/82	LOS ANGELES, CA	LOUISE TOUISSANT	(213) 478-0183
6/30/82	ORANGE COUNTY, CA	LOUISE TOUISSANT	(213) 478-0183

```
140 IFO(S)=6THEN150
145 PRINT"YOU HAVE NO ";C$:RETURN
150 PRINT"OK":O(S)=R:O=O-1:IFS=6THENO(6)=2
152 IFS=2THENPRINT"IT BREAKS":O(2)=0:IFO(6)=6THENO(6)=2:O=O-1
155 IFS<>6ANDS<>1THENRETURN
160 IFS=1THENPRINT"THEY SEEM DRY":RETURN
170 IF O(1)<>RTHENRETURN
176 PRINT"A HUGE PLANT GROWS  TO THE CEILING"
177 O(1)=0:O(6)=2:O(12)=R:RETURN
220 IFO(3)<>6THEN145
222 IFR=3ORR=4THEN245
225 IFC(2)THENPRINT"NO GOOD":RETURN
230 PRINT"THERE IS A LOUD RUM-BLE":C(2)=1:RETURN
245 IFC(1)=0THENPRINT"A BRIDGE APPEARS":C(1)=1:RETURN
246 PRINT"THE BRIDGE VANISHES":C(1)=0:RETURN
250 IFO(12)=3ANDR=3THEN260
255 PRINT"NOWHERE TO GO":RETURN
260 IFO(7)=6ANDO(9)=6ANDO(11)=6THEN262
261 PRINT"YOU DON'T HAVE ALL  THE TREASURE":RETURN
```

more ⟶

MACROS

. . . for the FORTH Assembler

I'm going to fill you in on a very powerful technique that can be used with the assembler in your AIM 65 FORTH system. I'll bet you didn't know that you could take often used assembly language code sequences and turn them into macros.

Let's look at the CODE sequence on page 12-14 of the AIM 65 Forth User Manual. If you don't have Forth at the moment, here's the routine as it's published:

```
CODE FORMAT XSAVE STX, 8A02 JSR, XSAVE LDX, NEXT JMP,
END-CODE
```

Assume that you have a number of other small code definitions and they all are a machine language subroutine, say on a disk I/O board, and have to save and restore the X register every time (since it's used as the data stack pointer).

We can easily define a macro to save and restore the X register and call a particular subroutine as follows.

```
ASSEMBLER DEFINITIONS
: CALL XSAVE STX, JSR,
        XSAVE LDX, ;
FORTH DEFINITIONS
```

The ASSEMBLER DEFINITIONS informs the system that you wish to add some new definitions to the assembler vocabulary.

The word CALL when referenced, will save X, do a JSR to whatever subroutine address is on the stack and then restore X when the program returns from the subroutine.

The FORTH DEFINITIONS sequence lets the system know that all further definitions will be compiled into the regular FORTH dictionary.

Now our FORMAT program can be rewritten to take advantage of the CALL macro.

```
CODE FORMAT 8A02 CALL NEXT JMP, END-CODE
```

That really tightens it up, doesn't it? The idea came from Charles Curley, one of the software types here at Rockwell. Quite powerful. This technique can, of course, be used for lots of other things than just a method for calling subroutines. New 6502 op codes can be written (such as CLA—clear Accumulator) and I/O ports can be initialized with one word. The possibilities are endless.

```
262 IFO(10)<>2ANDC(1)=0THEN266
263 PRINT"YOU DIDNT BLOCK THE PIRATE":O(7)=1:O(9)=1:O(11)=1
264 PRINT"TREASURES GONE!":O=O-3:RETURN
265 GOTO37
266 P=1135-M*3:PRINT" ":PRINT"YOU SCORED";P;"/1000":END
275 IFR=1ORR=2ORY$<>"CRO"ORC(1)=0THENPRINT"NICE TRY":RETURN
276 IFR=4THEN77
277 IFR=3THEN83
280 S=6:C$="WATER":GOTO100
285 S=6:GOTO135
290 IFO(4)=6THEN300
295 PRINT"YOU LACK WHATS NEED-ED":RETURN
300 IFR<>4THEN275
305 PRINT"THE STEAM CLEARS"
310 C(3)=1:O(9)=4:O(11)=4:O=O-1:O(4)=0:GOTO49
320 IF O(7)=6THENO(7)=2:O=O-1:PRINT"THE BATS GRAB YOUR  GOLD"
325 IFRND(1)>.9THENPRINT"SOMEONE IS WATCHING YOU"
330 GOTO49
```

COMMAND STRING CREATION FOR THE AIM 65/40

Jim Thompson
Rockwell International

The AIM 65/40 monitor command "&" (execute command string) allows the user to repeat a long sequence of commands with a minimum number of key strokes. The creation of a string of command characters from a printed listing may not be easy as the user needs to separate the key strokes from the characters output from the monitor/program. To aid the user, a program has been written to copy into a command string every key stroke used in a program sequence. The creation program is in three parts, initialization of the copy mode, termination of the copy mode, and the copy routine itself.

When the user is ready to start the creation of the string, he should enter the following:

{G}700 CR FROM=XXXX CR

All key strokes following this entry will be placed into memory, starting at the address given in reply to the "FROM=" prompt. No test is made on the length of the command string length, so the user should allow sufficient memory for the command string.

To terminate the copy of the key board input in to the command string, enter the following:

{G}075A CR (the address must be 4 characters)

This will restore the key input status to that prior to the enabling of the copy program. The command string may now be executed with the monitor command {&}.

Special operations such as 'MANUAL ENTRY' and 'SELF REPEATING' may be performed. To allow for a 'MANUAL ENTRY' of variable data within the command string (!!!), terminate the copy mode as described above at the point where manual entry is required. Record the address output on the display and then do the manual operation. When ready to resume the copying of the key board entries, execute the command string start program and respond to the prompt "FROM=" with the address saved from the above termination. Now continue with the command entry.

To create a command string which will, when it completes, start back at the beginning (an infinite loop) do the following. When ready to repeat, enter a memory display command of the start of the command string buffer.

{M}XXXX CR

Then do the normal copy mode termination. Record the address displayed. Then use this address minus 8 to display memory {M}. The byte displayed ($4B) should then be changed to a $26 ("&"). The string will then be programmed to repeat itself.

```
;LINKS TO SYSTEM
;
0800            VECTOR = $00FC
0800            IOVTAB = $0200
0800            COMIN  = $A334
0800            CSIO1X = $A6A0
0800            FROMX  = $AE35
0800            BLANK2 = $F377
0800            WRAX   = $F3BA
0800                   *=$0700


;************************************
;* ROUTINE TO INITIALIZE THE KEYBOARD
;* COPY AND LINK IT INTO THE SYSTEM
;************************************
;
;SAVE KEY INPUT VECTOR
;
0700 AD 00 02  SETCOP LDA IOVTAB
0703 8D 97 07         STA SYSVEC
0706 85 FC            STA VECTOR
0708 AD 01 02         LDA IOVTAB+1
070B 8D 98 07         STA SYSVEC+1
070E 85 FD            STA VECTOR+1
;
;COPY THE KEY INPUT JMP TABLE
;
0710 A0 08            LDY #8
0712 B1 FC     COPTAB LDA (VECTOR),Y
0714 99 99 07         STA JMPTAB,Y
0717 88               DEY
0718 10 F8            BPL COPTAB
;
;SAVE KEY INPUT GET VECTOR
;
071A AD 9D 07         LDA JMPTAB+4
071D 8D 4C 07         STA COPY+1
0720 AD 9E 07         LDA JMPTAB+5
0723 8D 4D 07         STA COPY+2
```

```
;SET GET VECTOR TO COPY ROUTINE
;
0726 A9 4B              LDA #<COPY
0728 8D 9D 07           STA JMPTAB+4
072B A9 07              LDA #>COPY
072D 8D 9E 07           STA JMPTAB+5
;
;GET ADDRESS OF COMMAND STRING BUFFER
;
0730 20 35 AE           JSR FROMX
0733 8E 4F 07           STX SAVE+1
0736 8C 50 07           STY SAVE+2
;
;SET THE FIRST CHARACTER IN THE STRING
;
0739 A9 21              LDA #'!'
073B 20 4E 07           JSR SAVE
;
;SET KEY INPUT VECTOR
;TO THE COPY JMP TABLE
;
073E A9 99              LDA #<JMPTAB
0740 8D 00 02           STA IOVTAB
0743 A9 07              LDA #>JMPTAB
0745 8D 01 02           STA IOVTAB+1
;
;RETURN TO THE MONITOR
;
0748 4C 34 A3           JMP COMIN


;************************************
;*  KEY COPY ROUTINE
;************************************
;
;COPY A KEY TO THE BUFFER
;
074B 20 FF FF   COPY    JSR $FFFF
;ADDRESS OF KEY INPUT GET
074E 8D FF FF   SAVE    STA $FFFF
;ADDRESS OF BUFFER
0751 EE 4F 07           INC SAVE+1
;STEP TO NEXT CELL
0754 D0 03              BNE COPEXT
0756 EE 50 07           INC SAVE+2
0759 60         COPEXT  RTS
```

```
;************************************
;* RESTORE NORMAL KEY INPUT AND
;* TERMINATE THE COMMAND STRING
;************************************
;
;RESTORE KEY INPUT GET
;
075A AD 97 07  RESTOR LDA SYSVEC
075D 8D 00 02         STA IOVTAB
0760 AD 98 07         LDA SYSVEC+1
0763 8D 01 02         STA IOVTAB+1
;
;REMOVE THE LAST 6 KEYS FROM THE
;STRING
0766 A2 05            LDX #5
0768 AD 4F 07  RESTO1 LDA SAVE+1
076B D0 03            BNE RESTO2
076D CE 50 07         DEC SAVE+2
0770 CE 4F 07  RESTO2 DEC SAVE+1
0773 CA               DEX
0774 10 F2            BPL RESTO1
;
;TERMINATE THE COMMAND STRING
;
0776 A9 21     TERMN8 LDA #'!'
0778 20 4E 07         JSR SAVE
077B 20 4E 07         JSR SAVE
;
;DISPLAY THE LINKING ADDRESS
;
077E 20 77 F3         JSR BLANK2
0781 AD 50 07         LDA SAVE+2
0784 AE 4F 07         LDX SAVE+1
0787 20 BA F3         JSR WRAX
;
;TERMINATE COMMAND STRING FOR POSSIBLE
;EDITOR USAGE
;
078A A9 0D            LDA #13
078C 20 4E 07         JSR SAVE
078F A9 00            LDA #0
0791 20 4E 07         JSR SAVE
0794 4C 34 A3         JMP COMIN



0797           SYSVEC *=*+2
0799           JMPTAB *=*+9
07A2                   END

 ERRORS=0000
```

Q&A

. . . on the R6651 ACIA

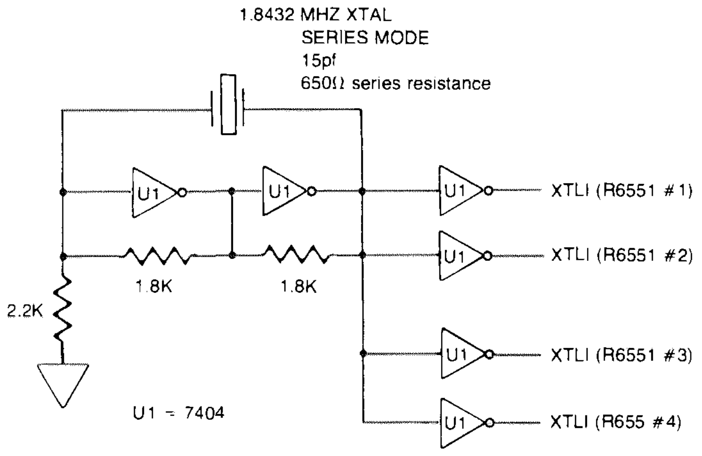
(EDITOR'S NOTE: This new column will be presenting the most often asked questions that our applications engineers have heard on certain Rockwell products. If you have a question on one of our products, drop me a line and it may get answered here if we feel it is of general enough interest. This month we thank Randy Dumse and Joe Hance, both of whom are applications engineers, for their help in compiling and answering these questions.)

What type of crystal do I need to use with the R6551 ACIA?

To drive just one device, use the following type of crystal: 1.8432 Mhz, series mode, 15pf, 650 ohm series resistance.

How do I drive more than one R6551 ACIA device from one crystal?

To drive more than a single ACIA device, you need to build an external crystal oscillator. The crystal just will not drive more than one input.



This circuit is identical to the one used on the RM 65 ACIA module and can drive up to two R6551 ACIA devices. To drive more than two XTLI inputs, simply add additional buffering as shown by the dotted lines.

If I'm not using the Data Carrier Detect (DCD) or Data Set Ready (DSR) lines in my particular application what should I do with them?

These two lines (DSR and DCD) should be tied to ground if you are not using the R6551 with a modem (or other device which needs these signals).

How do I get the ECHO mode on the R6551 ACIA device working properly?

To get the ECHO mode operational, set bits 2 and 3 of command register to 0 and bit 4 of that register to 1. The RTS output line will be low in the ECHO mode.

How do I properly initialize the R6551 ACIA device?

The first thing to do is perform a software reset to the R6551. This is done by writing something to the status register location (it doesn't matter what you write since what actually decodes the reset is the proper combination of register select lines and a low on the read/write line).

Next, make sure that bit 0 of the command register is set to 1. This sets the data terminal ready (DTR) line low and permits the ACIA device to function.

Then load the control register with the desired value. The R6551 ACIA is now ready to send and receive.

more ——►

VERIFYING A SUCCESSFUL RECORDING

by Dr. Lawrence A. Ezard
2149 Kentwood Dr.
Lancaster, PA 17601

After information from the text buffer has been recorded on cassette tape *a test should always be made to verify that a successful recording has been obtained* and that it will be possible later to read the information from the cassette tape to the text buffer using the text editor commands. The error detection capability during the reading of a file in the search mode operation is used. To test the information on cassette tape the verify procedure is as follows:

1. (a) Rewind the cassette tape and position the tape a couple inches or about five counts before the start of the desired file. This area of the tape should have nothing recorded on it. (Although a voice recording is permissible in this area no AIM-65 recorded data is permitted.)

more ——►

(b) Set the volume control to maximum and the tone control to maximum treble.

2. If remote control operation is being used turn off the cassette tape with the AIM-65 monitor command 1. Set the recorder to PLAY.

3. (a) Re-enter the text editor with the T command. The top line will be displayed and printed.

   (b) Position the line pointers to the bottom line using the B command. The last line will be displayed and printed.

   (c) Use the D command to move the line pointer down to the dummy line. The word END will be displayed.

4. The editor command R is now typed to obtain a pseudo read of the cassette tape information into the text buffer.

   A non-existent file name, such as DUMMY, will be used so that only a search operation, not a load operation for a file will be obtained.

An example of the above operations is shown with operator entries underlined.

```
<                      Comment: AIM-65 Monitor prompt
<T>                    Reenter the text editor and display the
TOP LINE               top line. Move the line pointer to dis-
=<B>                   play the last line in the text buffer.
LAST LINE              Move down one line to the dummy line.
=<D>                   The print key was pressed to print this
END                    line. Use R command to read multiple
=<R>                   lines of text into text buffer. Specify
IN=T F=DUMMY T=1       input from cassette tape, filename, and
                       remote control number 1.
```

The AIM-65 response to the R command is = <R>, a carriage return in the IN = prompt. The operator types T to indicate the information is coming from cassette tape. AIM-65 responds with the filename prompt F =. The operator types the non-existent filename DUMMY. AIM-65 responds with the tape recorder remote control prompt T =. The operator responds by typing 1 for remote control number 1.

5. The operator now types the RETURN key. If the remote control feature is used this will start the tape recorder. If the remote control feature is not being used the operator should manually start the tape recorder in the play operation mode.

6. (a) Typing the RETURN key initiates the start of a set of instructions which searches for the *start of a file* recorded on the cassette tape.

   (b) *When the first file is found* the search message will be displayed. If the file named PROG/ is read AIM-65 will display

   SRCH F=PROG1 BLK=XX

   where XX = the block count. The block count will change as the tape reading continues.

   (c) If there are any errors found in the tape recording as the search operation proceeds an error message will be displayed.

   (d) If the displayed block count is the last block count on the file, the displayed block count will not change until a new file is located.

   (e) If there are no error messages and the last block count is displayed then the tape recording verification operation is complete.

7. If the search message is not displayed as the tape is read, check the following possibilities.

   (a) Rewind the tape, play the tape, and listen to the tape to be sure that AIM-65 has recorded something on the tape.

   (b) Try steps 1 to 6 above again. Be sure to set volume and tone controls to maximum.

   (c) If the tape recorder play operation is started in a section of tape within a file (past the file name information stored on the tape) the search message will not be displayed. Instead, the block count will be displayed as part of the command message until the first file name is read. For example:

   IN=T F=DUMMY T=1 XX

   where XX is in the block count. The block count may change as the tape play continues.

   (d) If the first record, which contains the file name, was not successfully recorded on the cassette tape the AIM-65 response will be similar to 7(c) above. This can happen if the automatic level control on the tape recorder does not react fast enough at the beginning of the recording. It could also happen if the remote control switch does not close fast enough. In other words AIM-65 sends out the first record to be recorded but the tape recorder does not record the first record properly. One possible remedy is to increase the GAP value at A409 from $10 to $40 or $80. If this does not cure the problem the tape recorder head should be cleaned. And if this doesn't cure the problem a new tape recorder (possibly without automatic level control) should be tried.

TITLE PAGE PRINTER

C. M. Shaw
169 Rainbow Dr. N.W.
Ft. Walton Beach, FL 32548

For your line printer—

The following program is a 90-degree variation on the Banner Program published on page 15 of Interactive, issue number 2.

```
1000 REM TITLE PAGE PRINTER
1001 REM BY C. M. SHAW  22 DEC 1981
1002 REM DESIGNED FOR 8.5 BY 11 INCH PAPER
1003 REM 80 COLUMNS AND 6 LINES PER INCH
1004 REM END TEXT WITH "END$" IF LESS
1005 REM THAN SIX LINES.
1050 DIML$(8):L=1
1110 INPUT "HORZ CENTERING?, Y OR N";H$
1120 INPUT"VERT CENTERING? Y OR N";V$
1150 INPUT"TEXT";L$(0)
1200 IFL$(0)="END$" GOTO1400
1250 IFLEN(L$(0))>13 GOTO2150
1300 L$(L)=L$(0):L=L+1:IFL=7 GOTO2200
1350 GOTO 1150
1400 L=L-1
1401 BL=0:IFV$="N" GOTO1450
1403 AL=L*7:SL=54-AL:BL=INT(SL/(L+1))
1450 PRINT CHR$(5): REM * * *  ENABLE PRINTER
1500 FORLP=1TOL:LN=LEN(L$(LP)):MK=1
1555 IF BL=0 GOTO 1575
1560 FOR LB=1TOBL:PRINT:NEXT LB
1575 SP=INT(((13-LN)*6)/2)
1600 FOR R=1TO7:IFH$="N"GOTO1650
1605 IF LN=13 GOTO 1650
1610 FOR PS=1TO SP:PRINT" ";:NEXTPS
1650 FOR CH=1TOLN
1700 B=ASC(MID$(L$(LP),CH,1)):IFB>63 THEN B=B-64
1750 CO=62177
1800 FORCL=1 TO 5:P$=" "
1850 IF (PEEK(B+CO)ANDMK) THEN P$="#"
1900 PRINTP$;:CO=CO+64
1950 NEXT CL:PRINT" ";:NEXT CH:PRINT
2000 MK=MK+MK:NEXT R
2050 PRINT:PRINT:NEXT LP
2100 PRINT CHR$(1):END: REM DISABLE PRINTER * * *
2150 PRINT"REDO LINE ";L:GOTO 1150
2200 PRINT"FULL, READY TO PRINT?"
2250 GETQ$:IFQ$="Y"GOTO 1400
2300 IFQ$<>"N"GOTO2250
2350 GOTO 1050
```

DESIGNER'S COURSE

The Rockwell R6500 Microcomputer Designer's Course is . . .
A five-day course. $695 tuition includes complete documentation package, plus either an AIM 65 microcomputer (with an assembler) or a $300 discount off current price of a System 65 microcomputer development system.

R6500 Designer's School

Location	Week of	Location	Week of
Anaheim, CA	May 3	Chicago, IL	June 21
	June 7		August 9
	July 12		October 18
	August 9		December 13
	September 6		
	October 4	Dallas, TX	May 24
	November 8		July 26
	December 6		October 4
			December 6
Marlton, NJ	May 10		
	July 12		
	September 20		
	November 15		

The Rockwell PPS-4/1 Microcomputer Designer's Course is . . .
A five day course. $625 tuition includes complete documentation package, plus XPO-1 evaluation module. An outline of this course is available upon request.

TUITION MAY BE PAID BY CHECK, MONEY ORDER, MASTER CHARGE OR VISA CREDIT CARD, OR IT MAY BE BILLED TO YOUR COMPANY.

PPS-4/1 Designer's School

Contact School Location

Applications Engineering, D/832
Rockwell International
P.O. Box 3669,
Anaheim, CA 92803
PHONE: (714) 632-3860
TELEX/TWX: 910 591-1698

Rockwell International
5001B Greentree
Executive Campus, Rt. 73
Marlton, NJ 08053
PHONE: (609) 596-0090
TELEX/TWX: 710 940-1377

Rockwell International
10700 W. Higgins Rd.
Suite 102
Rosemont, IL 60018
PHONE: (312) 297-8862
TELEX/TWX: 910 233-0179

Rockwell International
Mail Code 417-100
Richardson, TX 75080
PHONE: (214) 996-6500
TELEX/TWX: 73307

"We can set up courses at your location"

AIM 65 PROGRAM EXCHANGE

Here's a good way to help increase the number of AIM 65 programs in the world. Jim Dantin is providing a clearinghouse of software from various sources. Send a S.A.S.E. for his information sheet.

Jim Dantin
1522 Springdale Dr.
Owensboro, KY 42301

(Continued from page 3)

converted is determined by the AIM 65 1MHZ $\phi 2$ clock. A complete scan through all channels requires 640 clock cycles i.e. scan rate = 1.56KHZ. The −12 to −15 volts power supply for the negative reference may already be available on some AIM systems, if not it must be provided for externally.

Simply reading locations 8000H to 8007H produces conversion data for the 8 analog input channels, for example the program below uses BASIC PEEK instructions to read and display the data from each channel as its corresponding channel number is operated on the keyboard.

```
10 REM============
20 REM 8 CHAN A/D
21 REM TEST PROG
30 REM    P. TOOMEY
40 REM============
50 REM
60 REM READS KEYBD
65 REM FOR CHANNEL
70 REM NUMBER AND
80 REM DISP DATA
90 REM A/D ADDRESS
95 REM IS 32768
96 REM
100 GETX$
110 IFX$<"0"THEN175
120 IF X$>"7"THEN175
130 X=VAL(X$)
140 FORDLY=1TO100
141 NEXTDLY
150 PRINT"CHANNEL NO:";
160 PRINTX;PEEK(32768+X)
170 GOTO 100
175 PRINT"NUMBERS 0-7 PLEASE"
177 PRINT
178 GOTO 100
180 END
```

(Continued from page 2)

Dear Sir,

You may be interested to learn of an application of the AIM-65 to machine tool control. I have three systems running where milling machines are fully controlled in all functions. A BASIC program performs the sorting of machining data while control of stepping motors (acceleration, slew, deceleration) is handled with machine language.

The purpose of this letter is two-fold, one is to give details on recording mixed language programs on tape and the recovery from tape with the minimum of key-strokes. This latter requirement is helpful when relatively unskilled labour is employed to operate the machines.

To dump a mixed programme, proceed as follows:

1) Escape to monitor and verify that the BASIC memory limits at 007F and 0080 are below the lowest machine language area.
2) Note the BASIC program limits XX, YY at 0075, 0076.
3) Start the normal dump procedure and answer the FROM=, TO= prompts as follows—

FROM=	TO=	
0	EA	
200	YY, XX	
a, a	b, b	
c, c	d, d	etc.

where a,a-b,b and c,c-d,d are machine language limits.

To load the program from a cold start, it is not necessary to activate BASIC, simply use the normal load procedure, enter BASIC with '6' and start the program with RUN.

This brings me to the second object of this letter to ask, is there a reverse USR function? Or can BASIC be entered and run with a single key stroke when in monitor?

May I add that your magazine has proved a most useful extension of the AIM-65 manuals which in themselves are invaluable.

Yours faithfully,
W. B. W. Alison
Alison Instruments
67 North Quay
Great Yarmouth
Norfolk, England
NR30 1JF

Dear Sir,

I wrote last week giving a method of dumping a mixed machine language/BASIC program, to allow loading without the necessity of first initialising BASIC. I enquired whether there was a method of entering BASIC straight into the run mode without the usual '6' RUN RET sequence.

Here is a method:—

1) Find 354 hex bytes RAM, (this can be in the BASIC work area or a file area which is later overwritten) and enter the appended program:—

```
2000                  ;INSTANT RUN BASIC
2000                  STORE  =$3000
2000                  ;OR SOME CONVENIENT
2000                  ;PLACE IN MEMORY.
2000                  ;THIS CAN BE IN
2000                  ;THE BASIC WORK AREA
2000
2000                          *=STORE+$300
3300  BA                      TSX
3301  86 A0                   STX $A0
3303  A0 00                   LDY #0
3305  84 AA                   STY $AA
3307  84 AB                   STY $AB
3309  A9 00                   LDA #<STORE
330B  85 AC                   STA $AC
330D  A9 30                   LDA #>STORE
330F  85 AD                   STA $AD
3311  B1 AA           PAGE    LDA ($AA),Y
3313  91 AC                   STA ($AC),Y
3315  C8                      INY
3316  D0 F9                   BNE PAGE
3318  E6 AD                   INC $AD
331A  E6 AB                   INC $AB
331C  A5 AB                   LDA $AB
331E  C9 03                   CMP #3
3320  D0 EF                   BNE PAGE
3322  A9 4C                   LDA #$4C
3324  8D 0F 01                STA $10F
3327  A9 32                   LDA #<RESET
3329  8D 10 01                STA $110
332C  A9 33                   LDA #>RESET
332E  8D 11 01                STA $111
3331  00                      BRK
3332
```

more

```
3332  A9 00      RESET  LDA #<STORE
3334  85 AC             STA $AC
3336  A9 30             LDA #>STORE
3338  85 AD             STA $AD
333A  A0 00             LDY #0
333C  84 AA             STY $AA
333E  B1 AC      RETN   LDA ($AC),Y
3340  91 AA             STA ($AA),Y
3342  C8                INY
3343  D0 F9             BNE RETN
3345  E6 AB             INC $AB
3347  E6 AD             INC $AD
3349  A5 AB             LDA $AB
334B  C9 03             CMP #3
334D  D0 EF             BNE RETN
334F  A6 A0             LDX $A0
3351  9A                TXS
3352  60                RTS
3353
```

2) Enter BASIC and make the direct command POKE 4, N : POKE 5, M : A = USR(0) : GOTO S where M and N are the pointers to the start of the new machine sub-routine (0 and 51 in the program attached) S is the first line number of your program.

3) Running halts at the BRK point and the program is now dumped as follows:—

Sections 0—3
10F—111
300—YY XX
Your original machine language areas.
The new machine language area.

YY.XX is the BASIC program limit found at pointers 76,75. If YY.XX is less 300 then this section is omitted.

4) When the program is loaded there is no need to initialise BASIC, simply load in the usual way, on completion key F2, when your program immediately starts. (The symbol <]> appears on the first line).

You may be interested in a couple of snapshots of AIM65 controlled CNC machine tools.

Yours faithfully,
W. B. Alison

Dear Editor:

After configuring our AIM-65 for current loop operation with a CRT, it was frequently necessary to switch back to the AIM keyboard, via the KB/TTY switch (S3), to obtain print-out on the AIM printer. This method of change-over soon became even more inconvenient with the addition of a new plastic case which made the KB/TTY switch nearly inaccessible. Our solution was to configure the AIM for software control of this switch.

The KB/TTY switch is a simple single-pole switch which either grounds pin 13 of Z32 (an R6522 VIA) for TTY operation, or lets this pin float high for the standard KB operation. What we did was to attach one end of a 560 ohm resistor to the PB0 pin of the applications connector (pin 9 of J1) and connect the other end, via a clip lead, to the upper contact of the KB/TTY switch (the one closest to the printer) which connects to Z32. If the KB/TTY switch is left open (KB), the status of bit PB0 will control the KB/TTY operation. The 560 ohm resistor protects PB0 if the KB/TTY switch is closed.

When the AIM is turned on, port B comes up as an input port, so that PB0 is high and the KB is selected. If PB0 is configured as an output and forced low, the CRT is selected. Although this can be done in any programming language, we have done it with the two FORTH words CRT and AIM shown below.

```
: CRT                          | : AIM
( SWITCHES CONTROL)            | ( RETURNS CONTROL)
( TO TTY AND SETS)             | ( TO AIM KB AND)
( BAUD RATE AT)                | ( PRINTER)
( 2400)                        | HEX
HEX                            | FF A002 C!
960 BAUD                       | FF A000 C! ;
FF A002 C!                     |
00 A000 C! ;                   |
```

The word CRT sets the baud rate, sets port B as an output port by loading its Data Direction Register (DDR) with ones, and sets PB0 to low. The baud rate, which is 960 hex (2400 decimal) in the example, may be adjusted by altering the number preceeding BAUD. The AIM word confirms the DDR data and sets PB0 high. If the other bits of port B are to be used for other purposes, these words should be rewritten accordingly to "OR" in the PB0 data.

Sincerely yours,
D. M. Gualtieri, and J. F. Ankner
Allied Corporation
Corporate Technology
P.O. Box 1021R
Morristown, New Jersey 07960

more ——►

Dear Sir:

The article by Mr. Ward in issue #7 of Interactive (p. 24), describing fast loading of Basic Programs via a machine language dump contains several problems. Be assured it will work, but it records many extra bytes that are not needed and which slow down the ultimate process of loading the program.

In the method espoused by Mr. Ward, he advocated recording of the entire zero page and the entire memory area allocated to Basic. As not all of the allocated area of Basic is ever used for storage of the program, and only a few bytes on zero page are used for Basic, this procedure can incur a dramatic increase in the number of blocks it takes to save a program to tape (thus resulting in a slower loading of the program than could be achieved). A better procedure is to exit to the monitor and examine memory locations 0073 to 0076 which contain in least significant/most significant order the start of Basic (0073, 0074) and the end of the Basic *Program* (0075, 0076). Now dump, in machine language format, locations 0073 to 007E which locations contain all the needed Basic pointers. The AIM will ask if you have more to dump at which point you instruct it to dump from the start of Basic (obtained in locations 73 and 74) to the end of the Basic Program (from locations 75 and 76). The computer now dumps only the Basic Program to tape and no unnecessary blocks are dumped.

To read back in the program enter and initialize Basic then exit to the monitor. Load the program with the "L" command and when finished loading type "G" and "RUN" and you are up and running your Basic Program. This method of loading and saving Basic programs can save up to 25% or more time than Mr. Ward's method.

I hope this improvement is found useful by the readership.

Sincerely,
John S. Wahlquist
1643 N. Formosa Ave., #4
Los Angeles, CA 90046

Dear Sirs,

I hope that the following little routine will be useful for AIM 65 users. This routine converts a 4-digit (Hex) binary number in its BCD representation, in a format convenient for output via the monitor output routines. If BCDBUF and HEXBUF are located in 0-page, it is fully relocatable and is only 31 bytes long.

```
         LDA #0
         LDX #2
CLEAR    STA BCDBUF,X  ;clear BCD buffer (3 bytes)
         DEX
         BPL CLEAR
         SED           ;set Decimal Mode
         LDY #16       ;load number of bits
NEXT     LDX #2        ;load number of BCD bytes -1
         ROL HEXBUF+1  ;shift left Hex number
         ROL HEXBUF    ;and load carry with MSB
ADDEC    LDA BCDBUF,X
         ADC BCDBUF,X  ;add BCD buffer to itself . . .
         STA BCDBUF,X  ;. . . including carry
         DEX
         BPL ADDEC
         DEY           ;Hex buffer scanning complete?
         BNE NEXT      ;no, push out next bit
         BRK           ;yes, conversion complete
```

This algorithm, well known by computer people, operates on a very simple basis: The BCD buffer is originally reset to all 0s, then the bits in the Hex buffer are shifted into the carry flag, sequentially, MSB first. At each iteration the BCD buffer is added to itself, IN DECIMAL, including the carry, that will be 0 or 1 according to the bit pushed out of the Hex buffer. In the next iteration the MSB of the Hex buffer, now part of the BCD buffer, will then be multiplied by 2, and the bit immediately following it in the Hex buffer will be added to the BCD buffer as is. At the end of the procedure, the MSB of the Hex buffer will have been multiplied by two 15 times, or by 2^{15} the bit following it by 2^{14} and so on, down to the LSB of the Hex buffer, that will have simply been added to the BCD buffer in the last iteration. This way every bit in the Hex buffer is multiplied by its own weight, but in decimal. This routine is especially useful in real-time control systems because of its fixed execution time.

Faithfully yours,
Antonio Silva M.S.E.E., P.E.
Via Anguissola 23 Milano 20146
Italy

Dear Eric,

Thanks again for all the great work you do for us AIM owners. I have had much character building trying to get the TTY Output program (INTERACTIVE No. 5, p. 13) to run.

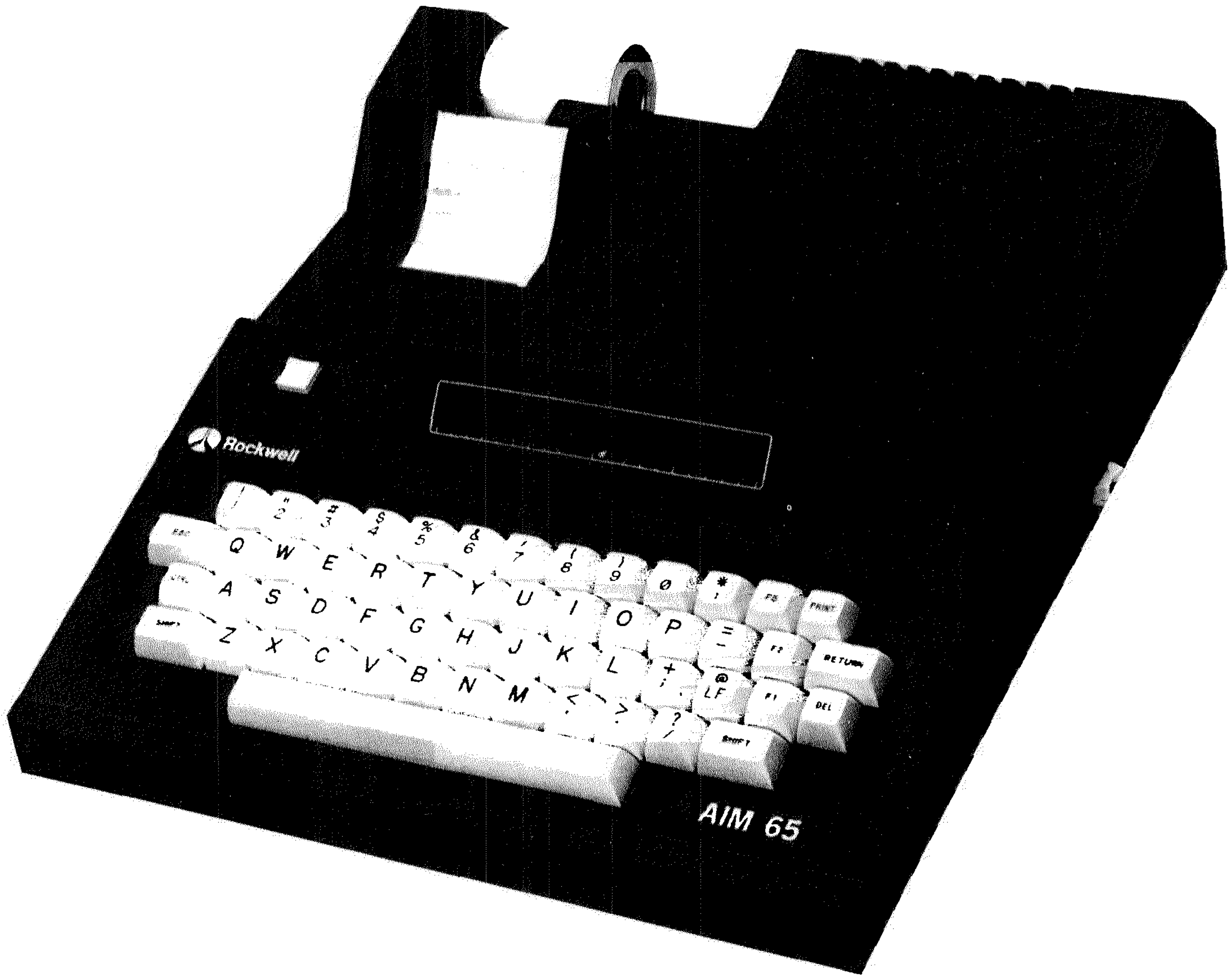
Here are the three problems I've had:

1) It would have been helpful to say the source code should be assembled on tape first. The article only told me what NOT to do.

2) CR = $8D NOT $0D

3) The location of the code means an automatic crash when BASIC is initialized because memory locations $200 are changed by BASIC. It is much better to locate the machine language subroutine at the upper end of memory i.e. $FD0. Then when BASIC is brought up memory is limited to 3534 for a 4K AIM. (4096-530-32 = 3534)

Yours sincerely,
Robert P. Barrett
Asst. Prof. of Engineering
and Computer Science.



A NEW CASE FOR AIM 65

Thought you'd be interested in seeing the new case/power supply combination that is now available.

The enclosure's ABS plastic top has a fine textured, non-reflective brown finish. The aluminum base includes mounting brackets for the AIM 65 and power supply, as well as line cord and circuit breaker. The enclosure measures 13" × 15.5" × 3.9".

The enclosure also includes an on/off switch, a pushbutton reset switch which mates with the AIM 65 reset switch, and is pre-wired with internal AC lines. Removable plugs in the cover allow access to the AIM 65 run/single step and KB/TTY switches. An external holder for the thermal printer's paper supply allows easy paper replacement without cover removal.

The regulated power supply which is supplied only with the enclosure provides +5V at 3 amperes and −24V at 0.5 amperes.

The enclosure with power supply, Model A65-006, has a retail list price of $165. Without the power supply, Model A65-002, the enclosure is $95. Quantity discounts are offered.

The new enclosure is available through Hamilton-Avnet and all other franchised AIM 65 distributors as well as many computer store dealers worldwide.

AIM 65 SPARE PARTS LIST

(EDITOR'S NOTE: This is an abbreviated spare parts list.
For a complete list of spare parts, see our OEM price list.)

Retail Price	Part Number	Description
$ 74.70	208R02-001	Thermal Printer
13.00	208R02-010	Print Head, Thermal Printer
29.25 /per segment	210R12-001	Display, Module (Intensity Code A, B, C, D, E)
5.00	29650N30-001	Manual, R6500 Programming
5.00	29650N31-001	Manual, R6500 Hardware
5.00	29650N36-001	Manual, AIM 65 USER's
5.00	29650N36-002	Manual, Monitor Program Listing
1.00	29650N50-001	Reference Card
1.00	29650N51-001	Summary Card
73.80	322R04-001	Keyboard Assembly, 54 Key
3.50 /box of 3	55201-001 (TT270)	Thermal Printing Paper (10+ boxes, $1.75 per box)
7.50	PA00-D020-001	Cable Assembly, 16 Conductor, AIM 65
3.85	PA00-D124-003	Filter, Display Assembly, AIM 65
2.00	PA00-D125-001	Shield, Static
1.50	PA00-D134-001	Tear Bar, Printer Paper
73.00	PA00-D200-001	Master Board Assembly (No Printer—No Display—No MOS) Soldered Components only with exchange of defective board
5.00	R2114-11	RAM
35.45	R3222-11	ROM, Monitor
35.45	R3223-11	ROM, Monitor
35.00	R3224-11	ROM, Assembler
32.50	R3225-11	ROM, Basic #2
32.50	R3226-11	ROM, Basic #1
35.00	R3298-12	ROM, PL/65 #1
35.00	R3299-11	ROM, PL/65 #2
32.50	R32J1-11	ROM, Forth #2
32.50	R32J2-11	ROM, Forth #1
6.95	R6502-11	CPU-Microprocessor
4.00	R6520-11	PIA, 40 PIN DIP
6.20	R6522-11	VIA
8.55	R6532-11	RIOT
200.00	A65-003	Service Test Board, AIM 65

- Minimum Retail Order $10.00
  F.O.B.-El Paso, Texas

† *Include State Tax*
Include $2.00 Handling Charge

NOTE: OTHER PARTS ARE AVAILABLE: For information call **714/632-2190 Anaheim, California or 800/351-6018 El Paso, Texas**

NEWSLETTER EDITOR
ROCKWELL INTERNATIONAL
P.O. Box "C"
4311 Jamboree Rd.
Newport Beach, CA 92660

Bulk Rate
U.S. POSTAGE
RATE
Santa Ana Calif.
PERMIT NO. 15