As you page through this first issue of the newsletter, you’ll notice that most of the articles have been written by Rockwell employees. Since the purpose of the newsletter is to provide you with a medium to exchange ideas with other AIM 65/6502 users, we’ll be looking forward to having an article from YOU (or even a comment about what you’d like to see) for the next issue.

You don’t need to be a professional writer to submit an article. We can smooth over and edit any rough spots there may be, as long as it’s readable. So please type it. We can also re-draw any diagrams that accompany your article. The best way to send assembly source listings is on cassette. Be sure to let us know if you’d like it returned. If you don’t have an assembler, we can accept handwritten source listings as long as they are easy to read and well commented — don’t forget to use labels for every referenced memory location.

I’ll look forward to hearing from you.

Best Wishes,

Eric C. Rehnke
Editor

To keep receiving this newsletter, subscribe now! The cost is $5 for 6 issues (or $8 overseas). As an incentive for charter subscriptions, we’ll send you the next 8 issues for $5 ($8 overseas) — that’s 2 additional issues free - if you subscribe now. This a one-time offer that will not be repeated. Just fill in the attached subscription request, add your check or money order payable to ROCKWELL INTERNATIONAL, and mail in the attached, postage paid envelope. (Payment must be in U.S. funds drawn on a U.S. bank.) No purchase orders.

All correspondence and articles should be sent to:
NEWSLETTER EDITOR
ROCKWELL INTERNATIONAL
P.O. Box 3669, RC55
ANAHEIM, CA 92803

AIM 65 SELF-TEST PROGRAM AVAILABLE

Rockwell is making the AIM 65 self-test program available through its spare parts facility. Order part numbers #EA74-M800 and #PL74-J100 for the Test Manual and Program Listing, respectively.

Refer to the spare parts list elsewhere in this newsletter for further information.

LOW COST PRINTER FOR AIM 65 (?)

An article in the February 1980 issue of MICROCOMPUTING page 186 (remember KILOBAUD?) explained the hows of interfacing a surplus Model 2970 Communications Terminal (Selectric-based) to a KIM or SYM.

The low price (the author paid $100 for his 2970) and the excellent print quality could offset the slow speed and complicated design of the Selectric mechanism for your application.

Since the SYM also uses a 6522 as its user I/O, conversion to AIM 65 would seem straightforward.

Although the author didn’t mention the printing speed of the Selectric, I understand it to be quite slow (around 10-15 characters per second). That works out to one fifth the speed of the DIABLO at about one thirtieth the price.

Hmmm . . . that’s not too bad.

COLOR GRAPHICS

That same issue of MICROCOMPUTING (Feb. 1980) also published the design of a low-cost video display which uses the AMI S68047 VDG.

The Video Display Generator operates in three modes: alphanumerics (32 x 16), semigraphics, or full graphics (up to 256 x 192 resolution).

Although the display chip was interfaced to an 8080 system in the article, an experienced AIM 65 user should have very little trouble in adapting the interface to his system.

The software must, of course, be completely re-written.
AIM 65 SPARE PARTS PROCUREMENT

Here's an abbreviated spare parts price list with some of the more commonly requested items. (All parts are available.)

For C.O.D. orders or inquiries, call: 800/351-6018.

Mail Orders should be directed to:
ROCKWELL INTERNATIONAL
SPARES CONTROL
P.O. BOX 3669, RC48
ANAHEIM, CA 92803

Add your state and city tax. On orders under $10.00, add $2.00 shipping and handling.

AIM SPARE PARTS LIST

<table>
<thead>
<tr>
<th>ROCKWELL PART NO.</th>
<th>DESCRIPTION</th>
<th>PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>208R02-001</td>
<td>THERMAL PRINTER</td>
<td>$74.70</td>
</tr>
<tr>
<td>208R02-010</td>
<td>PRINT HEAD</td>
<td>$13.00</td>
</tr>
<tr>
<td>341R29-001</td>
<td>RESET SWITCH (51)</td>
<td>$0.30</td>
</tr>
<tr>
<td>470R03-002</td>
<td>DARLINGTON TRANSISTOR ARRAY (Z21,Z32)</td>
<td>$2.71</td>
</tr>
<tr>
<td>TT270</td>
<td>THERMAL PRINTING PAPER 3 ROLLS/BOX</td>
<td>$3.50</td>
</tr>
<tr>
<td>PA00-D020-001</td>
<td>KEYBOARD TO AIM 65 CABLE</td>
<td>$7.50</td>
</tr>
<tr>
<td>PA00-D124-003</td>
<td>RED DISPLAY FILTER</td>
<td>$3.85</td>
</tr>
<tr>
<td>PA00-D125</td>
<td>DISPLAY ANTI-STATIC SHIELD</td>
<td>$2.00</td>
</tr>
<tr>
<td>PA00-D131-001</td>
<td>PAPER TEAR BAR</td>
<td>$0.73</td>
</tr>
<tr>
<td>PA00-D133-001</td>
<td>PAPER HOLDER</td>
<td>$7.13</td>
</tr>
<tr>
<td>R2114</td>
<td>RAM CHIP</td>
<td>$13.05</td>
</tr>
<tr>
<td>R3222</td>
<td>MONITOR ROM (Z22)</td>
<td>$35.45</td>
</tr>
<tr>
<td>R3223</td>
<td>MONITOR ROM (Z23)</td>
<td>$35.45</td>
</tr>
<tr>
<td>R6502</td>
<td>CPU (Z9)</td>
<td>$9.80</td>
</tr>
<tr>
<td>R6520</td>
<td>VIA (Z1mZ32)</td>
<td>$6.55</td>
</tr>
<tr>
<td>R6522</td>
<td>VIA (Z1mZ32)</td>
<td>$9.10</td>
</tr>
<tr>
<td>R6532</td>
<td>RIOT (Z33)</td>
<td>$12.95</td>
</tr>
<tr>
<td>EA74-M800</td>
<td>TEST MANUAL</td>
<td>$7.50</td>
</tr>
<tr>
<td>PL-EA74-1000</td>
<td>TEST PROGRAM LISTING</td>
<td>$7.50</td>
</tr>
<tr>
<td>210R12-001</td>
<td>4-DIGIT DISPLAY MODULE</td>
<td>$29.25</td>
</tr>
</tbody>
</table>

**CRT APP. NOTE INFO**

If you’ve been climbing the walls trying to find the Standard Microsystems CG-5004 character generator specified in our application note entitled “CRT MONITOR OR TV INTERFACE FOR AIM 65”, (#R6500N12) then listen up.

Standard Microsystems has discontinued that part and has replaced it with their CRT 7004. The newer part will retain for well under $20 and is available through their distributors. Contact the factory for more info:

STANDARD MICROSYSTEMS CORP.
35 Marcus Boulevard
Hauppauge, New York 11787
(516) 273-3100

**WARNING !!!**

Care must be taken so that the locations $A406 and $A407 in AIM 65 RAM are not accidentally altered by the user or his programs.

These locations ($A406 and $A407) contain the display linkage vector. Since the warm start sequence does not re-initialize this vector, once the locations are changed, pressing the RESET key will cause the machine to jump off into never-never-land. The only way to gain control is to turn the power off and then on again so AIM 65 can perform it’s cold start sequence. Turning off the power will, of course, cause any user programs to be lost.

**USING EPROMS IN AIM 65**

If you don’t have the optional BASIC or ASSEMBLER ROMS installed in your AIM 65, one or more EPROMS can be used. Two EPROMS which will plug in with no modification to the AIM are: the TMS 2516 and TMS 2532 from Texas Instruments and the Intel 2716.

The TMS 2532 (4K x 8 EPROMS) is a perfect match for AIM 65 because it occupies the same amount of address space as the Rockwell R2332 ROM. This allows programs to span two or more contiguous blocks of EPROM memory.

The TMS 2516 and Intel 2716, on the other hand, will occupy the lower 2K of AIM 65’s 4K per ROM slot. This is because All (address line 11) on AIM 65 will connect with CE (chip enable - pin 18) on the 2716/2516 and needs to be low to read from the EPROM device.

The Intel 2732 is not AIM 65-compatible because the functions of pins 18 and 21 (CE and A11) have been reversed.
FOR YOUR INFORMATION

No. I don’t mind if you also look elsewhere for AIM 65 information. Here are some other sources.

COMPUTE MAGAZINE
POB 5119
Greensboro, N. C. 27403

TARGET (newsletter)
c/o Donald Clem
R. R. No. 2, Conant Rd.
Spencerville, Ohio 45887

(Some publication is written almost entirely in German)

65xx MICRO MAG
Roland Lohr
Hansdorfer Str 4
2070 Ahrensburg
W. Germany

MICROCOMPUTING (formerly KILOBAUD)
Peterborough, NH 30458

Also, here is a list of application notes published by ROCKWELL for the 6502/AIM 65. They can be obtained for the asking from:

ROCKWELL INTERNATIONAL
MARKETING SERVICES
POB 3669, RC55
Anaheim, CA 92803

NOTE: This routine destroys the contents of the Symbol Table Starting Address Low and High ($0034 and $003B) and the Number of Symbols High and Low ($000B and $000C) so can only be used once per assembly. Of course, the program could be modified to transfer the data in these locations to other locations, but this is left up to the user.

AIM 65 SYMBOL TABLE ROUTINE

Sometimes it’s useful to obtain a symbol table from a assembly. Here is a short, fully relocatable routine that will do just that.

Simply install this program in some out-of-the-way spot (it now resides in the top page of a 4K AIM 65 system) and run it right after the assembly is done.

It’s handy to set the FI user Vector (locations $010C-$010E) to point to the start of the symbol table printing routine. This lets the FI key call for the symbol table printout.

Be sure to specify the document number and name.
EDIT BASIC PROGRAMS

I’ll bet you didn’t know that the AIM 65 text editor can be used to edit BASIC programs. Well, it can.

When you’re in BASIC and perform a SAVE to cassette - the program is saved in its ASCII format (not in the tokenized format as it’s stored in memory.) If you’ve ever had the printer on when you read in a BASIC program, you’ve seen how it’s saved on cassette.

There are three things you need to keep in mind, though, when you edit your programs:

1. Since the AIM 65 text editor limits the character per line count to 60, there can be no more than 60 characters per line in your BASIC program (BASIC normally permits 72 characters per line.) Any more than 60 characters will be ignored.

2. When BASIC programs are read into the editor, the first line of the text buffer will be blank. Leave this blank line in there or things will get fouled up.

3. When finished editing, go to the bottom of the text buffer with the ‘B’ key and drop down past the last line with the ‘D’ key. Next, press the ‘I’ key (for insert), followed by a Control ‘Z’ key down while pressing the ‘Z’ key), and then a ‘RETURN’ to terminate the insert.

The tape gap in location $A409 should be at least $20 to allow the BASIC interpreter time to interpret.

Now save the program to cassette using the editor ‘L’ (List) command after you have moved to the top of the buffer with the ‘T’ command.

CHECKSUM PROGRAM

Gordon Smith
Rockwell Hobby Computer Club

Here is a technique for verifying that your ROM’s are correct. The technique determines a check sum for each of the ROM’s or ROM pair. To make this easy to do, I am enclosing a check sum program which also could be used with some modification as a check sum subroutine.

The first section of the program uses the CRLF monitor subroutine to clear the display and the FROM and TO subroutines to get the starting and stopping addresses.

The second section of the program initializes the check sum to zero and also sets up a dummy third address byte for the start and stop addresses. The reason that the third address byte is used is to allow a proper ending of the checksum when the last address is FFFF.

The third section (starting at 032C) actually forms the running check sum by adding the currently addressed memory cell to the prior check sum.

The next section increments the start address until it equals the stop address + 1 as determined by the section starting at 0349. When the stop address is FFFF the incremental address must be 010000 at the time of termination. This is the reason for carrying the third address byte. If only two address bytes were used for the comparison, FFFF would increment to 0000 and the stop would never occur.

The final section uses the BLANK2 subroutine to space the display over so that the monitor prompt will not wipe out a digit of the result and then uses the NUMA subroutine three times to print out the three byte check sum.

The results of these check sums are as follows:

<table>
<thead>
<tr>
<th>BASIC CHIPS</th>
<th>B000 TO CFFF = OFC76B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B000 TO BFFF = 07CC47</td>
</tr>
<tr>
<td></td>
<td>C000 TO DFFF = 07FB24</td>
</tr>
<tr>
<td>ASSEMBLER CHIP</td>
<td>D000 TO DFFF = 071A67</td>
</tr>
<tr>
<td>MONITOR CHIPS</td>
<td>E000 TO FFFF = 0EB11B</td>
</tr>
<tr>
<td></td>
<td>E000 TO EFFF = 078675</td>
</tr>
<tr>
<td></td>
<td>F000 TO FFFF = 072AA6</td>
</tr>
</tbody>
</table>

The tape gap in location $A409 should be at least $20 to allow the BASIC interpreter time to interpret.
DATA FILES FOR AIM 65 BASIC

Ralph Reccia
Rockwell International

The ability to operate with data files greatly enhances the usefulness of AIM 65 BASIC.

The BASIC program listed here requires the use of two tape recorders and both must be in the remote control mode.

Note that lines 30 through 140 are an assembly language program which gets poked into memory locations $0F00 through $0F68 ($ = hexadecimal). Locations $0F69 through $0FFF are used to store the data files that are being used. Assembly language programs are inserted into BASIC programs as follows:

The assembly language program is assembled normally into its final destination address. Each data byte is then converted from a binary value to a decimal (BCD) value. (This part is a real drudge and could be made much easier with a utility program that does all this conversion automatically. Such things are simple for computers.) These decimal values are then put into our BASIC Program as a series of DATA statements (see lines 30-140). The program sequence in lines 10-20 is used to POKE these numbers into memory. The assembly language program is then accessed as a subroutine by the USR function (see Appendix FI in the AIM 65 BASIC manual.)

The user MUST limit the amount of memory available for BASIC to 3,839 bytes. This is done by entering 3839 in answer to "MEMORY SIZE?"

Subroutines 3000 and 3100 set up delays which allow the operator to perform the manual functions on the tape recorder and read the display as the program prompts the user as to functions that need to be performed. The program waits until the user responds that he is finished by use of the GET command. (See lines 1996, 3220, and 3330.)

The data file in the example program is a fixed record format type. There are four fields per record (see lines 1230 - 1260): the CATALOG # field, the AUTHOR field, the TITLE field, and the REMARKS field. These categories can easily be changed to suit whatever application you may have in mind. The number of fields per record can also be changed as long as you realize that in the present design, the data buffer length is 151 characters long ($EF69 - $EFFF).
However, all 151 characters are not available for user data. Some space is needed for element separation and an end-of-record character. Looking at lines 195, 196, and 219, you’ll see that the element separator is a semicolon (:) and the end of record is signified by an exclamation point (!). These characters are inserted at the appropriate points in the data buffer by the software and need not be entered by the operator. Furthermore, they cannot be used as data anywhere else in the file. Of course, these special characters can easily be changed by the user to any other convenient ASCII characters.

Each record needs one element separator per field and an end-of-record marker. To calculate the “actual” buffer space available to the user simply subtract the number of fields plus one (for the end of record character) from the total buffer length.

Our example file system has a 151-character buffer and four fields, so the user can enter 146 characters into the buffer, since

\[
151 - (4 + 1) = 146
\]

Advanced experimenters can change the size and location of the data buffer by changing the references in BASIC (lines 1040, 1290, 1320, and 2020) and the references to the label DATA contained in the assembly language subroutine.

Other routines can be added to do such things as search for and/or print various fields of the record, etc.

By the way, the end of file indicator consists of a colon (:) as the first character followed by an exclamation point (!).
THE FOLLOWING SUBROUTINES DESCRIBE A METHOD OF WRITING AND READING FILES WHILE EXECUTING A BASIC PROGRAM ON AIM 65. THE ROUTINES MAY BE RELOCATED.

OUTALL=$E9BC

CRLF=$E9F0

TAOSET=$F21D

OUTTAP=$F24A

TAISET=*EDEA

GETTAP=i:EE29

; THIS ROUTINE OUTPUTS DATA TO TAPE

LDA #*B7 ; SET PB7 TO AN OUTPUT

STA SA802

JSR TAOSET SET RECORDER 2 AS OUTPUT

LDA #*0C TURN RECORDER 2 OFF

STA $A800

RTS

THE FOLLOWING ROUTINE READS THE TAPE

LDA #0 INITIALIZE ACR

NO JSR TAISET, SET TAPE 1 FOR INPUT

SYNC JSR GETTAP READ CHAR FROM TAPE TO ACC

CMP BEQ # # " YES

CMP BNE BEQ #*16 NO SYNC YES

LDA STA RTS #*0C $A800 TURN RECORDER 1 OFF

; THIS ROUTINE OUTPUTS DATA TO ADD

LDX AGAIN LDA INC

GETMOR JSR SAVE ACC

CMP BEQ LINFD, CHECK FOR RECORD SEPARATOR

CMP BNE DONE, CHECK FOR END OF FILE

JSR OUTALL OUTPUT ACC TO ADD

LINFD JSR CLEAR ADD

DONE JSR CRLF

CRLF

END OF ASSEMBLY
A COUPLE OF 6522 APPLICATIONS

Conrad Boisvert
Synertek, Inc.

6522 - GENERATING LONG TIMED INTERVALS

The 6522 Versatile Interface Adapter contains two 16-bit counter/timers for a variety of purposes, among them the generation of timed interrupts. Each counter is 16 bits long, so the maximum count-down is $2^{16}$ or 65,536 counts. With a 1 MHz processor clock rate, this translates to a maximum time of about 54.4 msec.

In some cases, this may not be long enough. To achieve longer timed intervals, several schemes may be used. Among them are:

1. Increment or decrement a memory location each time the timer interrupt occurs. In this way, an additional factor of up to 256x can be achieved, resulting in a maximum of about 16.8 seconds. However, extra program steps are needed.

2. The two 6522 timers may be connected externally (Figure 2), resulting in an effective 32-bit counter/timer. In this way, intervals longer than one hour may be achieved.

To cascade the two counters together, it is necessary to do the following:

1. Connect PB6 and PB7 together. These pins will not be usable as general I/O functions in this case.

2. Program T1 mode to free-run with output on PB7.

3. Program T2 mode to count pulses on PB6 input.

In this way, the waveform on PB7 is:

Since timer T2 pulse-counting mode counts negative-edge transitions, it is clear that T2 will decrement as follows:

Thus, T2 decrements will occur at the following intervals:

$$T_{2 \text{ RATE}} = 2x (T_1 \text{ COUNT}) \times (02 \text{ PERIOD})$$

And, hence, the total time will be,

$$T = 2x (T_1 \text{ COUNT}) \times (T_2 \text{ COUNT}) \times (02 \text{ PERIOD})$$

Thus, the maximum is $2 \times 65,425 \times 65,536 \times 1 \text{ us} = 8590 \text{ seconds} = 142 \text{ minutes} = \text{about 2-1/2 hours}$.

6522 -GENERATING A 1 Hz SQUAREWAVE SIGNAL

The 6522 (Versatile Interface Adapter) has two integral 16-bit timers intended to perform a variety of programmable functions. One capability is to use timer T1 to generate continuous squarewave output on peripheral pin PB7.

The timer is clocked by the system clock, $\phi 2$, which normally operates at 1 MHz. The waveform generated is illustrated in Figure 1.

To cascade the two counters together, it is necessary to do the following:

1. Connect PB6 and PB7 together. These pins will not be usable as general I/O functions in this case.

2. Program T1 mode to free-run with output on PB7.

Note that the period of the waveform is $2N + 4$ cycles, with a 16-bit counter, the maximum number of cycles is where N is the number set into the timer.

$$N_{\text{MAX}} = 16 \times 1 = 65,535$$

Hence, the maximum programmable period is:

$$P_{\text{MAX}} = 2N_{\text{MAX}} + 4 = 131,074 \text{ cycles}$$
This is about 131 msec for a 1 MHz system clock, considerably less than 1000 msec, the period for a 1 Hz signal.

One way to extend the period is to use the PB7 output signal as a clock input to the shift register on the 6522. If a pattern of 11110000 is set into the shift register, then the output of the shift register will appear as Figure 2.

![Shift Register Output Waveform](image)

Note that the period is extended by a factor of 8 by this method.

\[ P_{\text{max}} = 8(2N+4) \]

Hence for 1 Hz, \( P_{\text{max}} = 1,000,000 \) and \( N = 62,498 \). Thus, it is necessary to store the number 62,498 into the timer T1 in order to generate the 1 Hz waveform. Then translated into hexadecimal format, the result is F422, and F4 is loaded into the high byte and 22 into the low. The step-by-step sequence for programming this is shown in Figure 3.

Note especially the following points:

* Loading the T1 high-order counter (Register 5) initiates the timer in its free-running mode.

* PB7 data direction must be set to an output for the pulses to occur.
BASIC REAL TIME CLOCK
Mark Reardon
ROCKWELL INTERNATIONAL

Here's a machine language program converted to data statements that gets 'poked' into high memory from Basic. This particular program doesn't include the capability for displaying the time — that must be added by the user if needed.

Don't forget to limit the memory size to 4045.

There are plenty of remarks throughout the program so there's no need for a really detailed explanation.

```plaintext
85 REM THIS PROGRAM WRITTEN BY MARK REARDON
10 REM THIS IS A 24 HOUR CLOCK PROGRAM WRITTEN FOR
20 REM BASIC ON THE AIM 65. IT UTILIZES THE USER
30 REM 6522'S CLOCK ONE. THE FIRST SIX LINES OF CODE
40 REM STORE THE INTERRUPT CLOCK ROUTINE IN UPPER MEMORY.
50 REM WHEN INITIALIZING BASIC LIMIT MEMORY TO 4045.
60 FOR I=1 TO 50: READ X: POKE 4045+I, X: NEXT I
70 DATA 72, 138, 72, 230, 223, 166, 223, 224, 16, 208, 32
80 DATA 169, 0, 133, 223, 230, 222, 162, 60, 228, 222, 208, 20
90 DATA 133, 221, 230, 221, 228, 221, 208, 12
100 DATA 133, 221, 230, 220, 162, 24, 228, 220, 208, 2
110 DATA 133, 220, 104, 170, 173, 4, 160, 104, 64
120 REM THE NEXT TWO LINES OF CODE ENTER THE TIME INTO
130 REM THE COUNTERS. TO CHANGE THE START TIME INSERT
140 REM NEW VALUES IN THE FIRST THREE ENTRIES OF THE DATA
150 REM STATEMENT. THEY ARE HOURS, MINUTES, AND SECONDS.
160 FOR I=1 TO 4: READ X: POKE 219+I, X: NEXT I
170 DATA 0, 0, 0, 0
180 REM SET UP THE INTERRUPT ENABLE AND THE
185 REM AUXILIARY CONTROL REGISTERS
190 POKE 40974, 192: POKE 40971, 64
200 REM SET UP IRQ VECTOR TO CLOCK PROGRAM.
210 POKE 41984, 206: POKE 41985, 15
220 REM LOAD AND START TIMER ONE.
230 POKE 40964, 34: POKE 40965, 244
240 H=PEEK(220): M=PEEK(221): S=PEEK(222)
250 REM NOW THE TIME IS H HOURS, M MINUTES, AND S SECONDS.
260 REM TO ADJUST THE CLOCK THE VALUES IN 40964 (FINE) AND
270 REM 40965 (COARSE) CAN BE CHANGED. LARGER VALUES
280 REM SLOW DOWN THE CLOCK.
```
TTY TIP

From the Editor

At terminal speeds of 2400 baud and above, AIM 65 has a hard time figuring the correct baud rate. These baud rate values must be entered manually from the AIM 65’s keyboard.

Set the KB/TTY switch to KB and press the RESET button (this gets you back in the keyboard mode). You should now see the monitor prompt "<" on the LED display. Press the "M" key and examine location $A417. The display should now be displaying the contents of four memory locations starting with $A417. Press the "/'" key to modify memory followed by the correct baud rate values (two bytes) as found in Section 9.2.3 of the AIM 65 USER’S GUIDE. Now press the return key on the AIM 65 keyboard. Next, set the KB/TTY switch to TTY and press the space bar on AIM 65. If your terminal was set to the same baud rate as you set up in AIM 65, you should see the monitor prompt on your terminal which signifies you’re up and running in the terminal mode.

GENERAL PURPOSE REMOTE CONTROL INTERFACE

If you use several different cassette recorders with AIM 65 and have to change the polarity of the remote control cable each time you change recorders, then you’ll appreciate this little goodie!

This relay will enable AIM 65 to control most any cassette unit regardless of control signal polarity.

---

NEWSLETTER EDITOR
ROCKWELL INTERNATIONAL
P.O. Box 3669, RC55
Anaheim, CA 92803 U.S.A.