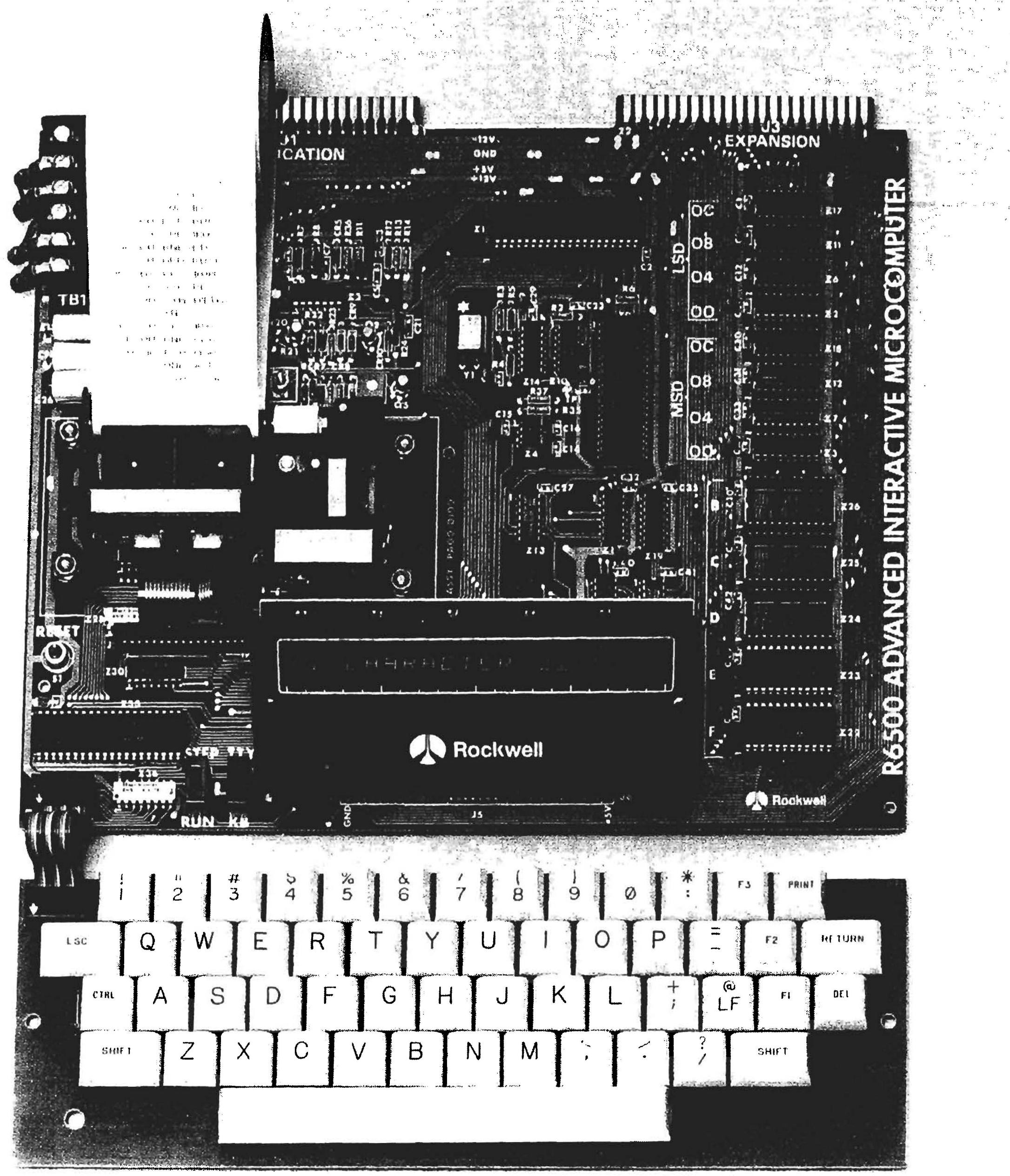
INTERACTIVE

ISSUE NO. 1



Rockwell International

...where science gets down to business

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

COPYRIGHT 1980 ROCKWELL INTERNATIONAL CORPORATION

Rockwell does not assume any liability arising out of the application or use of any products, circuit, or software described herein: neither does it convey any license under its patent rights nor the patent rights of others. Rockwell further reserves the right to make changes in any products herein without notice.

AIM 65 SELF-TEST PROGRAM AVAILABLE

Rockwell is making the AIM 65 self-test program available through it's 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 adpating 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

ROCKWELL PART NO.	DESCRIPTION	PRICE
208R02-001	THERMAL PRINTER	$74.70
208R02-010	PRINT HEAD	13.00
341R29-001	RESET SWITYCH (S1)	.30
470R03-002	DARLINGTON TRANSISTOR ARRAY (Z21,Z30)	2.71
TT270	THERMAL PRINTING PAPER 3 ROLLS/BOX	3.50
PA00-D020-001	KEYBOARD-TO-AIM 65 CABLE	7.50
PA00-D124-003	RED DISPLAY FILTER	3.85
PA00-D125	DISPLAY ANTI-STATIC SHIELD	2.00
PA00-D131-001	PAPER TEAR BAR	73
PA00-D133-001	PAPER HOLDER	7.13
R2114	RAM CHIP	13.05
R3222	MONITOR ROM (Z22)	35.45
R3223	MONITOR ROM (Z23)	35.45
R6502	CPU (Z9)	9.80
R6520	PIA (U1)	6.55
R6522	VIA (Z1mZ32)	9.10
R6532	RIOT (Z33)	12.35
EA74-M800	TEST MANUAL	7.50
PL-EA74-J100	TEST PROGRAM LISTING	7.50
210R12-001	4-DIGIT DISPLAY MODULE (MUST SPECIFY INTENSITY CODE WHICH IS FOUND ON LOWER LEFT HAND CORNER OF MODULE. THE CODE WILL BE A LETTER (A THROUGH E) OR A COLOR (RED THROUGH WHITE)	29.25

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 EPROPM 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

MICRO
POB 6502
So. Chelmsford, Mass. 01824

MICROCOMPUTING (formely KILOBAUD)
Peterborough, NH 30458

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

(This publication is written almost entirely in German)

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

Document No.	Document Name
223	R6502/R6532 Timer Interrupt Precautions
224	System 65 to AIM 65 Interface
230	RS-232C Interface for AIM 65
231	Intferfacing R6500 Microprocessors to a Floppy Disk
235	Interfacing KIM-4 to AIM 65
247	Using KIM-1 Tapes with AIM 65
238	A CRT Monitor or TV Interface for AIM 65
241	Preparing an AIM 65 Basic Program for PROM/ROM Operation.

Be sure to specify the document number and name.

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 F1 user Vector (locations $010C-$010E) to point to the start of the symbool table printing routine. This lets the F1 key call for the symbol table printout.

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.

```
0000                    ;THIS SYMBOL TABLE
0000                    ;PRINTING ROUTINE
0000                    ;WAS WRITTEN BY
0000                    ;ERIK SHOVGAARD OF
0000                    ;DENMARK
0000                    ;IT WILL MAKE ONLY
0000                    ;ONE LISTING OF
0000                    ;THE SYMBOL TABLE
0000                    ;PER ASSEMBLY.
0000                    ;PRESS THE 'F1' KEY
0000                    ;AFTER THE ASSEMBLY
0000                    ;TO GET A LISTING OF
0000                    ;THE TABLE.
0000                           *=$010C
010C  4C 00 0F                 JMP SYM
010F                    LNK    =$3A
010F                           *=$F00
0F00  20 71 E8          SYM    JSR $E871
0F03  20 F0 E9                 JSR $E9F0
0F06  A0 00             NEW    LDY #0
0F08  B1 3A             SYMLP  LDA (LNK),Y
0F0A  20 BC E9                 JSR $E9BC
0F0D  C8                       INY
0F0E  C0 06                    CPY #6
0F10  D0 F6                    BNE SYMLP
0F12  A9 3D                    LDA #'=
0F14  20 BC E9                 JSR $E9BC
0F17  A9 24                    LDA #'$
0F19  20 BC E9                 JSR $E9BC
0F1C  B1 3A                    LDA (LNK),Y
0F1E  20 46 EA                 JSR $EA46
0F21  C8                       INY
0F22  B1 3A                    LDA (LNK),Y
0F24  20 46 EA                 JSR $EA46
0F27  20 F0 E9                 JSR $E9F0
0F2A  18                       CLC
0F2B  A9 08                    LDA #8
0F2D  65 3A                    ADC LNK
0F2F  85 3A                    STA LNK
0F31  A9 00                    LDA #0
0F33  65 3B                    ADC LNK+1
0F35  85 3B                    STA LNK+1
0F37  38                       SEC
0F38  A5 0C                    LDA $0C
0F3A  E9 01                    SBC #1
0F3C  85 0C                    STA $0C
```

```
0048  0F3E  A5 0B      LDA $0B
0049  0F40  E9 00      SBC #0
0050  0F42  85 0B      STA $0B
0051  0F44  05 0C      ORA $0C
0052  0F46  D0 BE      BNE NEW
0053  0F48  60         RTS
0054  0F49              END
```

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" (hold the "CTRL" key down while pressing the "Z" key), and then a "RETURN" to terminate the insert.

The tape gap in locaion $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:

BASIC CHIPS	**B000**	**TO**	**CFFF**	**= OFC76B**
	B000	**TO**	**BFFF**	**= 07CC47**
	C000	**TO**	**DFFF**	**= 07FB24**
ASSEMBLER CHIP	**D000**	**TO**	**DFFF**	**= 071A67**
MONITOR CHIPS	**E000**	**TO**	**FFFF**	**= 0EB11B**
	E000	**TO**	**EFFF**	**= 078675**
	F000	**TO**	**FFFF**	**= 072AA6**

(CHECKSUM PROGRAM CONT'D)

INITIALIZE 'F1' KEY
010C 4C JMP 0300

PICK UP START AND STOP ADDRESSES

```
010C  4C  JMP  0300
0300  20  JSR  E9FO
0303  20  JSR  E7A3
0306  AD  LDA  A41C
0309  85  STA  01
030B  AD  LDA  A41D
030E  85  STA  02
0310  20  JSR  E83E
0313  20  JSR  E7A7
0316  AD  LDA  A41C
0319  85  STA  04
031B  AD  LDA  A41D
031E  85  STA  05
```

CLEAR 3RD BYTE TEST ADDRESSES AND CHECK SUM

```
0320  A0  LDY  #00
0322  84  STY  00
0324  84  STY  03
0326  84  STY  06
0328  84  STY  07
032A  84  STY  08
```

FORM CHECK SUM

```
032C  18  CLC
032D  B1  LDA  (01),Y
032F  65  ADC  08
0331  35  STA  08
0333  A9  LDA  #00
0335  65  ADC  07
0337  85  STA  07
0339  A9  LDA  #00
033B  65  ADC  06
033D  85  STA  06
```

INCREMENT ADDRESS TO PICK UP NEXT VALUE

```
033F  E6  INC  01
0341  D0  BNE  0349   06
0343  E6  INC  02
0345  D0  BNE  0349   02
0347  E6  INC  00
```

TEST FOR LAST TERM

```
0349  A5  LDA  04
034B  C5  CMP  01
034D  A5  LDA  05
034F  E5  SBC  02
0351  A5  LDA  03
0353  E5  SBC  00
0355  B0  BCS  032C   D5
```

PRINT FINAL RESULTS AND RE-ENTER MONITOR

```
0357  20  JSR  E9F0
035A  20  JSR  E83B
035D  A5  LDA  06
035F  20  JSR  EA46
0362  A5  LDA  07
0364  20  JSR  EA46
0367  A5  LDA  08
0369  20  JSR  EA46
036C  4C  JMP  E1A1
```

```
FROM=B000          TO=BFFF
  07CC47
<[>
FROM=C000          TO=CFFF
  07FB24
<[>
FROM=D000          TO=DFFF
  071A67
<[>
FROM=E000          TO=EFFF
  078675
<[>
FROM=F000          TO=FFFF
  072AA6
<[>
FROM=B000          TO=CFFF
  0FC76B
<[>
FROM=E000          TO=FFFF
  0EB11B
```

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 F1 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 semi-colon (;) 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 one 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 text 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 dummy record with a colon (:) as the first character followed by an exclamation point (!).

```
1 REM BASIC FILE HANDLING PROGRAM FOR AIM 65
2 REM THIS ROUTINE DESCRIBES A METHOD OF
3 REM SAVING AND LOADING DATA DURING THE
4 REM EXICUTION OF A BASIC PROGRAM
5 REM
6 REM ROCKWELL INTERNATIONAL 8/24/79
7 REM
8 REM SET ASSEMBLY ROUTINES TO LOC $0F00
9 REM
10 FORI=0TO104 READX
20 POKE3840+I,X NEXT
30 DATA169,183,141,2,168
40 DATA32,29,242,169,35,32,74,242,162,0
50 DATA189,105,15,32,74,242,232,201,33,208,245
60 DATA169,12,141,0,168,96,169,0,141,11,168
70 DATA32,234,237,32,41,238
80 DATA201,35,240,6,201,22,208,242,240,243
90 DATA162,0,32,41,238,157,105,15,232
100 DATA201,33,208,245,169,12,141,0,168,96
110 DATA162,0,189,105,15,232,201,59,240,10
120 DATA201,33,240,12,32,188,233,76,74,15
130 DATA32,240,233,76,74,15,32,240,233
140 DATA32,240,233,96
189 REM
190 REM SET TAPE GAP TO $30, TURN BOTH
191 REM REMOTE CONTROLS OFF,SET UPPER
192 REM BYTE OF USER VECTOR TO $0F AND
193 REM SET TAPEOUT $A435 TO TAPE 2
194 REM NOTE DEFAULT IS TAPE 1
195 REM '!' SYMBOL IS END OF RECORD
196 REM ';' SYMBOL IS ELEMENT SEPARATOR
197 REM
200 POKE41993,48:POKE43008,12:POKE5,15:POKE42037,1
210 ER$="!" ES$=";"
500 PRINT"DO YOU WANT TO 1)" GOSUB3000
510 PRINT"UPDATE A FILE 2)":GOSUB3000
520 PRINT"CREATE A NEW FILE" GOSUB3000
530 PRINT"3)READ A FILE":GOSUB3000
540 INPUT"INPUT 1,2OR3";X
550 ONXGOTO1000,1200,2000
1000 GOSUB3200
1010 GOSUB3300
1020 PRINT"CASSETTE 1 TO PLAY":GOSUB3100
1030 PRINT"CASSETTE 2 TO RECORD" GOSUB3100
1040 POKE4,32:X=USR(Y):A=PEEK(3945)
1044 REM
1045 REM CHECK FOR END OF FILE
1046 REM
1050 IFA=58THEN1220
1060 POKE4,72:X=USR(Y)
1070 PRINT"WANT TO KEEP IT":GOSUB3000
1080 INPUT"TYPE Y OR N",A$
1090 IFA$="N"THEN1040
1100 POKE4,0:X=USR(Y):GOTO1040
1200 GOSUB3300
1210 PRINT"CASSETTE 2 TO RECORD":GOSUB3100
1220 POKE4,0
1230 INPUT"CATALOG #";CN$
1240 INPUT"AUTHOR";AU$
1250 INPUT"TITLE";TI$
1260 INPUT"REMARKS";RE$
1262 REM
1263 REM
1264 REM
1265 REM
1266 REM PUT THE ELEMENTS TOGETHER
1267 REM
1270 A$=CN$+ES$+AU$+ES$+TI$+ES$+RE$+ES$+ER$
1280 FORI=1TOLEN(A$)
1290 POKE3944+I,ASC(MID$(A$,I,1)):NEXT
1300 X=USR(Y) INPUT"MORE Y OR N";A$
1310 IFA$="Y"THEN1230
1320 POKE3945,58:POKE3946,33:X=USR(Y)
1330 GOSUB3200
1340 PRINT"WISH TO READ TAPE":GOSUB3000
1345 INPUT"TYPE Y OR N";A$
1350 IFA$="Y"THEN1990
1360 GOSUB3200
1370 PRINT" TYPE RUN TO RESTART":END
1990 PRINT"CHANGE TAPE TO UNIT":GOSUB3000
1993 PRINT"ONE TYPE D WHEN DONE":GOSUB3000
1996 GETA$:IFA$<>"D"THEN1996
2000 GOSUB3200
2010 PRINT"CASSETTE 1 TO PLAY":GOSUB3000
2020 POKE4,32:X=USR(Y):A=PEEK(3945)
2030 IFA=58THEN2100
2040 POKE4,72:X=USR(Y):GOTO2020
2100 PRINT"END OF FILE" GOSUB3000
2110 GOTO1360
3000 FORI=1TO1500:NEXT:RETURN
3100 FORI=1TO2500:NEXT:RETURN
3200 PRINT"CASSETTES TO REWIND":GOSUB3100
3210 POKE43008,60:PRINT"TYPE D WHEN DONE":GOSUB3000
3220 GETA$:IFA$<>"D"THEN3220
3230 POKE43008,12:RETURN
3300 PRINT"ADVANCE TAPES SO YOU":GOSUB3000
3310 PRINT"ARE PAST LEADER":GOSUB3000
3320 POKE43008,60
3325 PRINT"TYPE D WHEN DONE":GOSUB3000
3330 GETA$:IFA$<>"D"THEN3330
3340 POKE43008,12:RETURN
```

```
0000               ;BASIC FILE HANDLER ASSEMBLY ROUTINES 8/24/79
0000               ;        R. RECCIA
0000               ;
0000               ;THE FOLLOWING SUBROUTINES DESCRIBE A METHOD OF
0000               ;WRITING AND READING FILES WHILE EXECUTING A
0000               ;BASIC PROGRAM ON AIM 65. THE ROUTINES MAY BE
0000               ;RELOCATED
0000               ;
0000               ;
0000               OUTALL=$E9BC
0000               CRLF=$E9F0
0000               TAOSET=$F21D
0000               OUTTAP=$F24A
0000               TAISET=$EDEA
0000               GETTAP=$EE29
0000               ;
0000                       *=$0F00
0F00               ;
0F00               ;THIS ROUTINE OUTPUTS DATA TO TAPE
0F00               ;
0F00   A9 B7               LDA #$B7        ;SET PB7 TO AN OUTPUT
0F02   8D 02 A8            STA $A802
0F05   20 1D F2            JSR TAOSET      ;SET RECORDER 2 AS OUTPUT
0F08   A9 23               LDA #'#'        ;OUTPUT # AS BEGINNING OF FILE
0F0A   20 4A F2            JSR OUTTAP
0F0D   A2 00               LDX #0
0F0F   BD 69 0F    MORE    LDA DATA,X      ;LOAD ACC WITH CHARACTER
0F12   20 4A F2            JSR OUTTAP      ;OUTPUT ACC TO TAPE
0F15   E8                  INX
0F16   C9 21               CMP #'!'        ;COMPARE TO TOTAL STRING
0F18   D0 F5               BNE MORE
0F1A   A9 0C               LDA #$0C        ;TURN RECORDER 2 OFF
0F1C   8D 00 A8            STA $A800
0F1F   60                  RTS
0F20               ;
0F20               ;THE FOLLOWING ROUTINE READS THE TAPE
0F20               ;
0F20   A9 00               LDA #0
0F22   8D 0B A8            STA $A80B       ;INITIALIZE ACR
0F25   20 EA ED    NO      JSR TAISET      ; SET TAPE 1 FOR INPUT
0F28   20 29 EE    SYNC    JSR GETTAP      ;READ CHAR FROM TAPE TO ACC
0F2B   C9 23               CMP #'#'        ;CHECK IF BEGINNING OF FILE
0F2D   F0 06               BEQ YES
0F2F   C9 16               CMP #$16        ;CHECK FOR SYNC CHARACTER
0F31   D0 F2               BNE NO
0F33   F0 F3               BEQ SYNC
0F35   A2 00       YES     LDX #0
0F37   20 29 EE    GETMOR  JSR GETTAP      ;INPUT CHARACTER
0F3A   9D 69 0F            STA DATA,X
0F3D   E8                  INX
0F3E   C9 21               CMP #'!'        ;CHECK FOR END OF FILE
0F40   D0 F5               BNE GETMOR
0F42   A9 0C               LDA #$0C        ;TURN RECORDER 1 OFF
0F44   8D 00 A8            STA $A800
0F47   60                  RTS
0F48               ;
0F48               ;THIS ROUTINE OUTPUTS DATA TO AOD
0F48               ;
0F48   A2 00               LDX #0
0F4A   BD 69 0F    AGAIN   LDA DATA,X
0F4D   E8                  INX
0F4E   C9 3B               CMP #';'        ;CHECK FOR RECORD SEPARATOR
0F50   F0 0A               BEQ LINFD
0F52   C9 21               CMP #'!'        ;CHECK FOR END OF FILE
0F54   F0 0C               BEQ DONE
0F56   20 BC E9            JSR OUTALL      ;OUTPUT ACC TO AOD
0F59   4C 4A 0F            JMP AGAIN
0F5C   20 F0 E9    LINFD   JSR CRLF
0F5F   4C 4A 0F            JMP AGAIN
0F62   20 F0 E9    DONE    JSR CRLF
0F65   20 F0 E9            JSR CRLF
0F68   60                  RTS
0F69               ;
0F69               ;
0F69   41          DATA     BYTE 'A'
0F6A                        END

ERRORS = 0000 <0000>
END OF ASSEMBLY
```

A COUPLE OF 6522 APPLICATIONS NOTES

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.

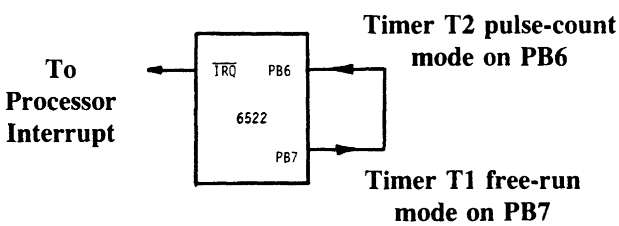


Figure 1 - Connection to Use T1 and T2 as 32-bit Counter

PROGRAMMING CONSIDERATIONS

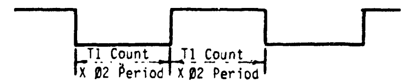
To cascade the two counters together, it is necessary to do the following:

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

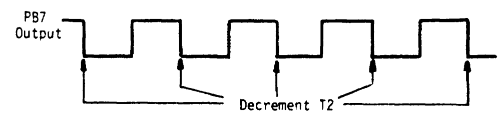
2. Program T1 mode to free-run with output on PB7.

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:

T2 RATE=2x (T1 COUNT) x (02 PERIOD)
And, hence, the total time will be,
T=2x (T1 COUNT) x (T2 COUNT) x (02 PERIOD)

Thus, the maximum is 2 X 65,425 X 65,536 X 1 us = 8590 seconds = 142 minutes = 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, ϕ 2, which normally operates at 1 MHz. The waveform generated is illustrated in Figure 1.

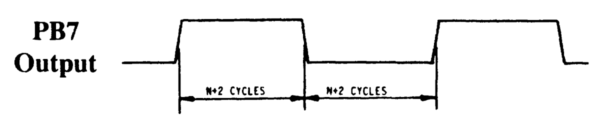


Figure 1 - PB7 Output Waveform

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.

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

Hence, the maximum programmable period is:

$P_{MAX} = 2N_{MAX} + 4 = 131,074$ 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 sift register on the 6522. If a pattern of 11110000 is set into the shift register, then the output of the sift register will appear as Figure 2.

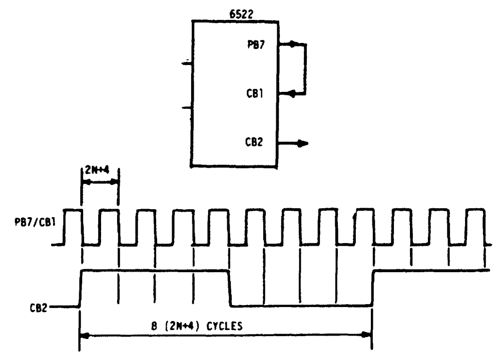


Figure 2 - Shift Register Output Waveform

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

$$P_{MAX} = 8\,(2N+4)$$

Hence for 1 Hz, P_{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.

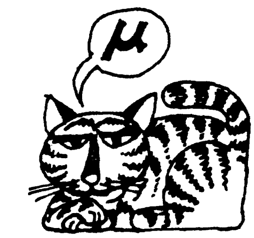
* PB7 data direction must be set to an output for the pulses to occur.

```
2000                        ;PROGRAM TO GENERATE 1HZ SQUARE-WAVE
2000                        ;OUTPUT ON R6522 PB7 OUTPUT PIN USING
2000                        ;T1 TIMER AND SHIFT REGISTER.
2000                        ;
2000                        ;---------------R6522 ADDRESSES-----------
2000                        ;
2000                        DDRB    =$A002          ;DATA DIRECTION REG.
2000                        T1CH    =$A005          ;T1 COUNTER HIGH BYTE
2000                        T1LL    =$A006          ;T1 LATCH LOW BYTE
2000                        SR      =$A00A          ;SHIFT REGISTER
2000                        ACR     =$A00B          ;AUX CONTROL REG
2000                        ;
2000                                *=$0200         ;START ADDRESS
0200                        ;
0200    A9 F0                       LDA #%11110000
0202    8D 0A A0                    STA SR          ;STORE SHIFT PATTERN
0205    A9 DC                       LDA #$DC
0207    8D 0B A0                    STA ACR         ;SETUP T1 AND SHIFT REG
020A    A9 22                       LDA #$22
020C    8D 06 A0                    STA T1LL        ;LOW BYTE
020F    A9 F4                       LDA #$F4
0211    8D 05 A0                    STA T1CH        ;HIGH BYTE AND START IT
0214    A9 80                       LDA #$80
0216    8D 02 A0                    STA DDRB        ;SET PB7 = OUTPUT
0219    4C 19 02            LOOP    JMP LOOP        ;STOP HERE
021C                                .END
```

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.

```
 5 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 FORI=1TO50:READX:POKE4045+I,X:NEXTI
70 DATA72,138,72,230,223,166,223,224,16,208,32
80 DATA169,0,133,223,230,222,162,60,228,222,208,20
90 DATA133,222,230,221,228,221,208,12
100 DATA133,221,230,220,162,24,228,220,208,2
110 DATA133,220,104,170,173,4,160,104,64
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 FORI=1TO4:READX:POKE219+I,X:NEXTI
170 DATA 0,0,0,0
180 REM SET UP THE INTERRUPT ENABLE AND THE
185 REM AUXILIARY CONTROL REGISTERS
190 POKE40974,192:POKE40971,64
200 REM SET UP IRQ VECTOR TO CLOCK PROGRAM.
210 POKE41984,206:POKE41985,15
220 REM LOAD AND START TIMER ONE.
230 POKE40964,34:POKE40965,244
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

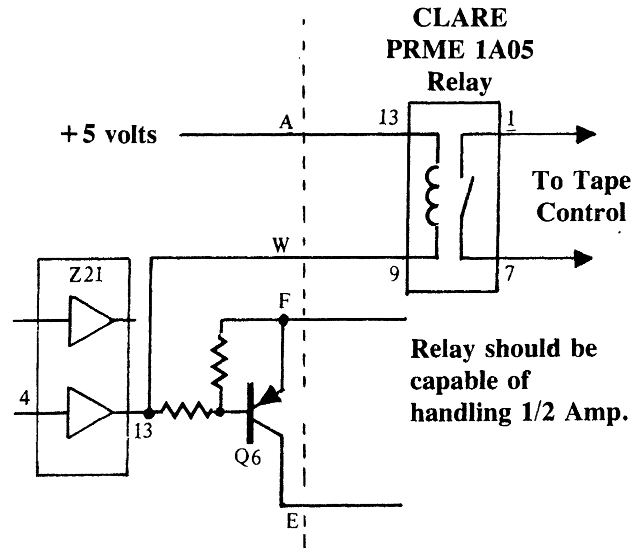
At terminal speeds of 2400 baud and above, AIM 65 has a hard time figuring the correct baud rate. These baud rate values must be entered manually from the AIM 65's keyboard.

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.

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