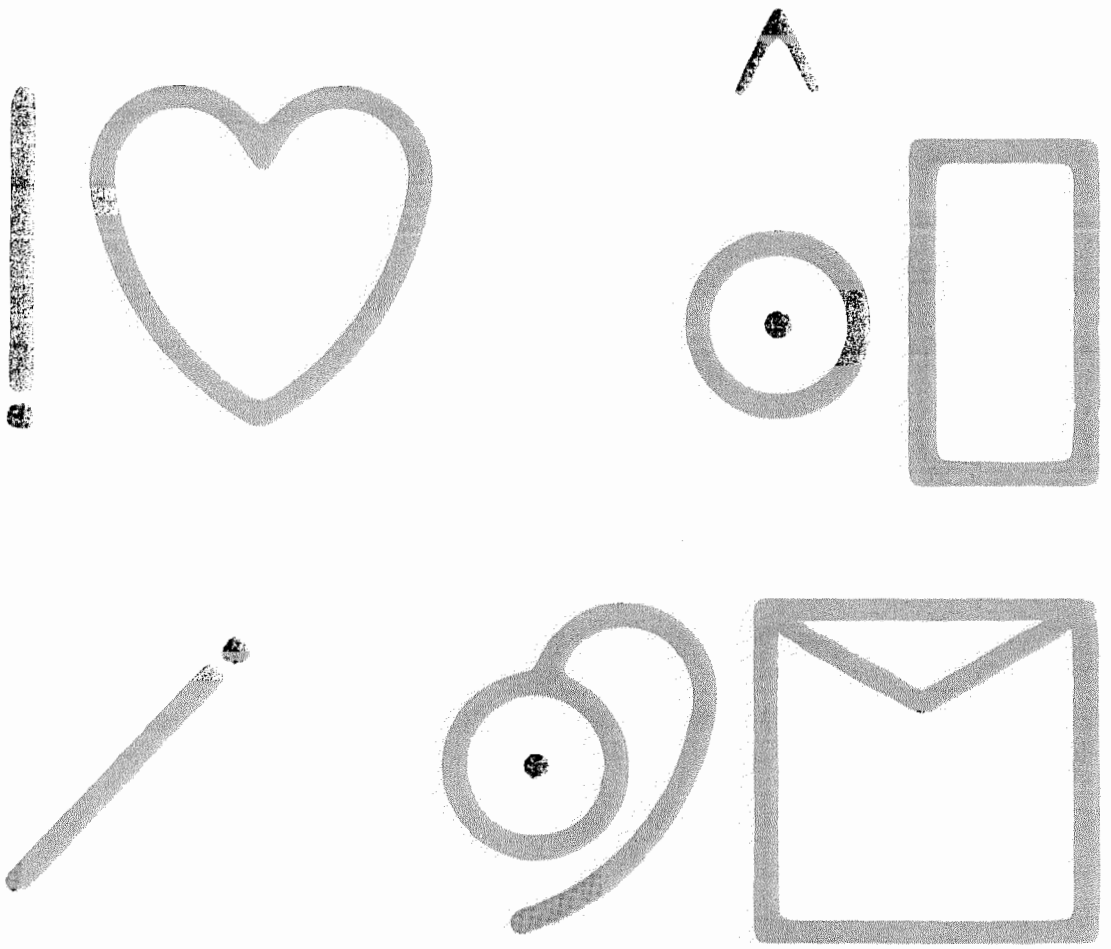
INTERACTIVE

ISSUE NO. 4



EDITOR'S CORNER

THIS MONTH'S COVER . . .

. . . has a message directed to those of you who can understand Blissymbolics (a graphics-based communications system intended for non-speaking persons). The message means "please read this newsletter." See the article on page 3 for more information on Blissymbolics and how your AIM 65 can be used with this graphics based language.

APP. NOTE UPDATE

Remember the application note I mentioned in the last issue (PRINTER CONTROL WITH THE R6522 VIA R6500 N21) which showed how to interface a low-cost printer mechanism to the R6522? Well, we've recently been informed by the company that makes the mechanism that there have been some changes to the units that could require some changes to the software driver routines in our applications notes. If you are planning to use one of their printer mechanisms, be aware that they have changed them and now have new model numbers. Better contact them for more information.

Two Day Corporation
Executive Mart
203 E. Main
Riverton, WY 82501

AIM 65 REPAIR CHARGES

Effective immediately the flat rate charge for out-of-warranty repairs on the AIM 65 will be $49.80.

In cases where there is extensive damage to the machine, as when the power has been hooked up incorrectly, the flat rate charge is not used. Instead, an estimate is sent to the customer for approval.

Follow the procedures outlined in the AIM 65 User Manual for returning your unit for repair.

Editor

FOR YOUR INFORMATION

AIM 65/ MICROPRODUCTS APPLICATIONS ENGINEER

(714) 632-0975—Use this number when you have technical questions concerning the AIM 65 system or are having difficulty interfacing to the AIM 65.

DEVICE APPLICATIONS ENGINEER

(714) 632-3860—Use this number when you have technical questions concerning individual 6500 family devices whether or not they are on the AIM 65.

SERVICE INFORMATION

(800) 351-6018—Call this number when your AIM 65 is broken and needs repair. Their address is:
AIM 65 REPAIR
Rockwell International
6 Butterfield Trail Dr.
El Paso, TX 79924

LITERATURE & DISTRIBUTOR/DEALER INFORMATION

(714) 632-3729, (800) 854-8099 (in California call (800) 422-4230)—Call one of these numbers when you need literature for a certain product, information on your nearest Rockwell dealer/ distributor or to request a particular application note.

SALES INFORMATION

(714) 632-3698—Call this number when you need price information for AIM 65 or Microflex 65 accessories or other Rockwell products.

SPARE PARTS

(714) 632-2190—Call this number when you want to order spare parts for your AIM 65. (The minimum cash order is $10.)

To keep receiving this newsletter, subscribe now! The cost is $5 for 6 issues ($8 overseas). *(NO CASH OR PURCHASE ORDERS WILL BE ACCEPTED)* (Payment must be in U.S. funds drawn on a U.S. bank).

All subscription correspondence and articles should be sent to:

EDITOR, INTERACTIVE
ROCKWELL INTERNATIONAL
POB 3669, RC 55
ANAHEIM, CA 92803

COMMUNICATIONS FOR THE HANDICAPPED

. . . using the AIM 65

Sam Caldwell
Director of Habilitation Engineering Services
Northwest Louisiana State School

(EDITOR'S NOTE: Here's a great example of how high technology can be used to make life easier for the handicapped community. This article was presented as a paper at the International Conference of Rehabilitation Engineering, and is being reprinted with the permission of the publishers. It was written by Sam Caldwell, Director of Rehabilitation Engineering Services at the Dept. of Health and Human Resources at the Northwest Louisiana State School.

In further conversations with Mr. Caldwell, he pointed out that even though most of the work being done to support the handicapped has been funded by the government, and is therefore public domain, very little in the way of technical information has been published to aid others in their work. I would like to commend Sam on his openness and hope that others in this field take the hint and start letting us in on all the work that is being financed with our tax dollars.)

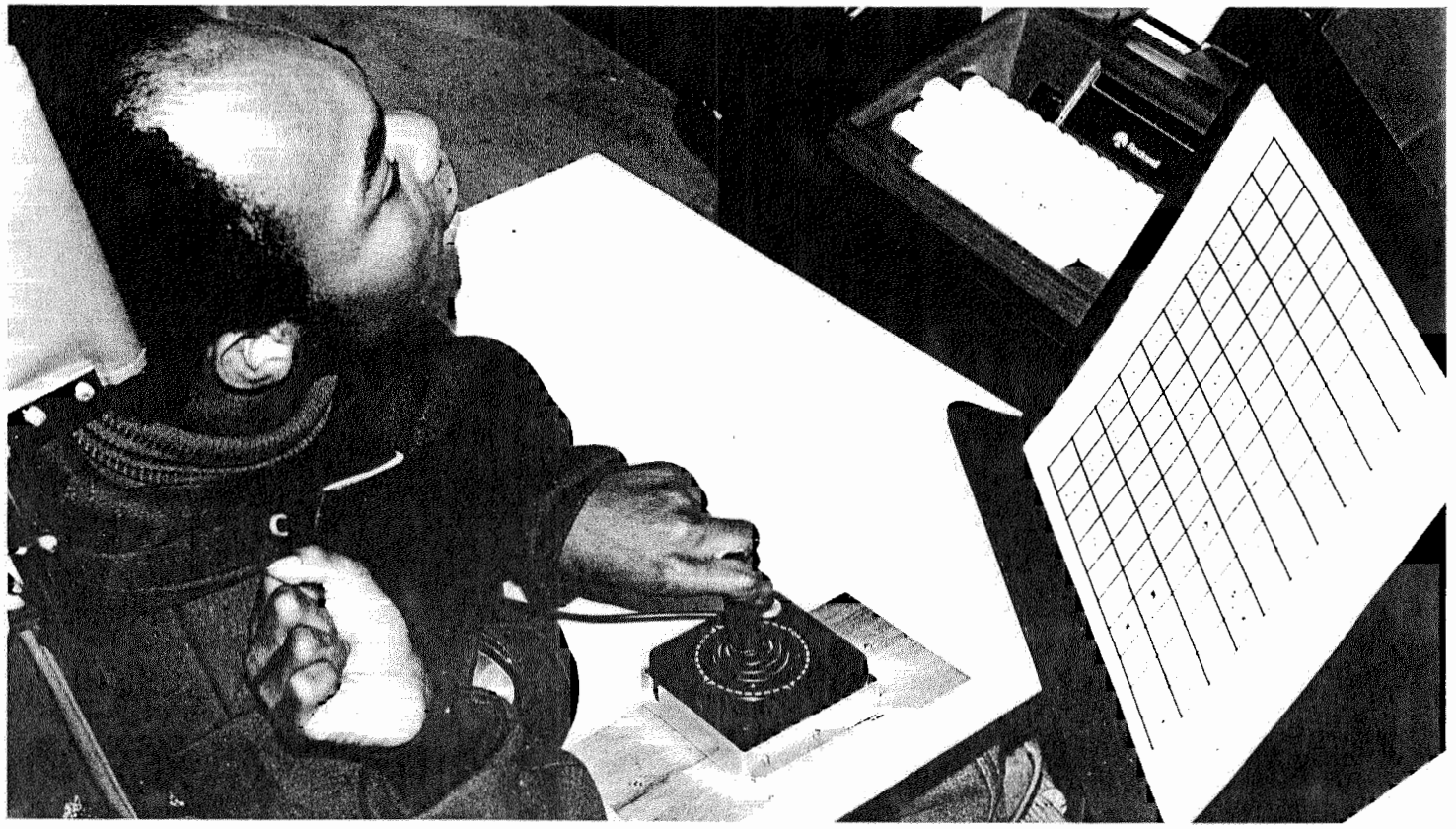
ABSTRACT

An inexpensive microcomputer based prosthetic communication system designed specifically to meet the needs of functionally non-verbal physically handicapped users is described. Computer/"real world" interfacing and BASIC program are explained. Educational and recreational benefits of microcomputer systems are examined and future plans outlined.

BACKGROUND

The microcomputer revolution has provided the handicapped with an exciting and extremely powerful new tool. Today a sophisticated computer system readily adapted for the handicapped user can be purchased for less than $700.00. Affordable microprocessor based "intelligent" consumer products are appearing in increasing numbers and hold tremendous promise for reducing dependency. A major frustration facing today's habilitation/rehabilitation engineer is finding enough time to explore and keep up with current technologies. Cost and availability are in most cases no longer top priority concerns.

The Northwest Louisiana State School Habilitation Engineering Department is investigating the potential of microcomputers for multiply handicapped, severely and profoundly retarded persons. As of this writing, all efforts have focused on using the Commodore PET 2001-8 and Rockwell AIM 65 microcomputers. Both machines are the personal property of a school employee and most programs and hardware modifications have been developed independently of state employment.



In the photo, Norman Potts, a 32-year-old resident of the Northwest School is using the AIM 65 to compose a message. Norman is nonverbal, retarded and physically handicapped. He has learned over 50 symbols and words using the AIM 65 with Blissymbolics and has become quite skilled in playing a target practice game.

The full typewriter-style keyboard, 20 character alphanumeric LED display, dual cassette recorder interface, on-board 20 column thermal printer, 8K BASIC and no-fuss interfacing make the AIM 65 an excellent candidate for habilitation/rehabilitation applications. A total package system complete with enclosure, power supply and basic ROM can be had for as little as $600.00.

INTERFACING

Most habilitation/rehabilitation applications require the addition of external switches tailored to the physical abilities of the handicapped user and necessitate real-world/computer interfacing. The AIM 65 and PET computers both have extremely flexible interface hardware on-board making child's play of what can easily become a stumbling block when working with other machines. All connections to the AIM 65 are made via its user-dedicated 6522 versatile interface adapter chip (VIA). A minimum configuration requires only two connections (See Figure 1). Sound requires adding two more wires and an inexpensive speaker-amplifier. If the control switch exposes the handicapped user to possible electrical contact with the computer, a simple battery powered optical isolator or reed relay can be inserted between the control switch and 6522 input/output port (See Figure 2).

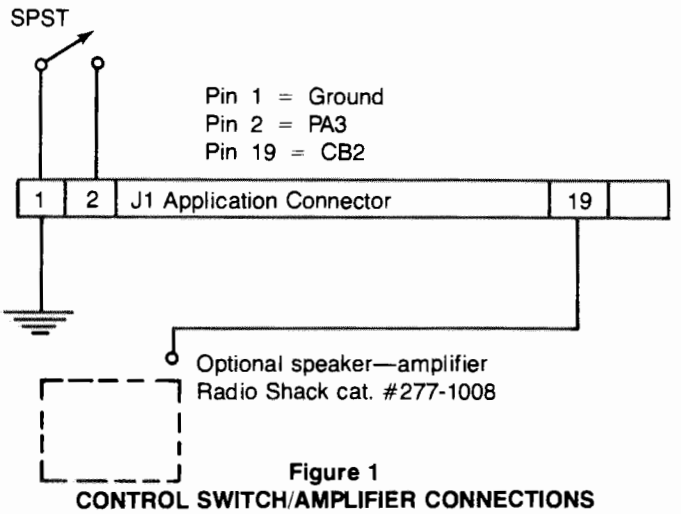


Figure 1
CONTROL SWITCH/AMPLIFIER CONNECTIONS

At the conclusion of this article is a listing of an AIM 65 program designed to enhance communication for speech-handicapped persons. The system consists of an AIM 65 microcomputer with 4K of RAM memory, 8K ROM BASIC, power supply, 44 pin edge connector, compatible cassette tape recorder, momentary contact SPST switch, battery powered speaker-amplifier and Blissymbolics Communication Foundation, 10 × 10, 100 vocabulary Bliss board. (Blissymbolics Communication Institute, 350 Rumsey Rd., Toronto, Ontario, Canada M4G 1R8). Other communication boards and vocabularies may, of course, be used providing responses can be identified via vertical and horizontal coordinates and the vocabulary listed in data statements starting at line 1000 are replaced accordingly.

PROGRAM OPERATION

Operation is simple and straightforward. When the program is run, the computer responds with "SCAN RATE?". The number entered in response to this query will determine the speed at which vertical and horizontal coordinates are displayed. The larger the number the slower the scan rate. Values between 50 and 60 have proved workable for most of our physically handicapped users. Once the scan rate has been entered and the RETURN key pressed, the numbers one through ten are alternately displayed on the left side of the LED panel and a "beep" is emitted through the attached speaker amplifier.

If, for example, the user wished to communicate the word "help", he would first locate the corresponding symbol and activate the control switch when the appropriate numeric and alphabetic coordinates are displayed, i.e., 7,D. Therefore, when the number 7 is displayed, the user momentarily closes the control switch. The computer responds by emitting a high-frequency "beep" and beginning sequential display of the letters A–J. When D is presented, the user again closes the control switch. The computer generates a short tone signaling recognition of his selection, displays the English equivalent of the symbol in the approximate center of the LED panel and resets to numeric scanning. If the user wishes to print a displayed word, he activates the control switch when either the 9,A or 10,A coordinates are displayed. In the special case of "help" an auditory alert is sounded until the user again presses the control switch.

The basic coordinate scanner program is being expanded to provide enhanced editing and print formatting capabilities. Future plans include the development of a rechargeable battery power supply and a rugged enclosure suitable for wheelchair mounting.

As of this time, two functionally non-verbal residents, classified as either profoundly or severely retarded and physically handicapped, have learned to use the AIM 65 communication scanner. The computer generates an immediate translation and written record of the user's responses. Early observations suggest that instantaneous translation of Bliss into English reinforces the learning of English. The potential of this system as an educational tool beyond establishing viable communications appears great.

In addition to practical uses in communication and education, the AIM 65 and PET computers have been very well received as a source of entertainment. The flexibility and accessibility of these machines allow the development of both individual and group games which can accommodate a wide range of physical and mental limitations. The computer serves as an equalizer—making it possible, in some cases for the first time, for handicapped persons to play games with one another without outside assistance.

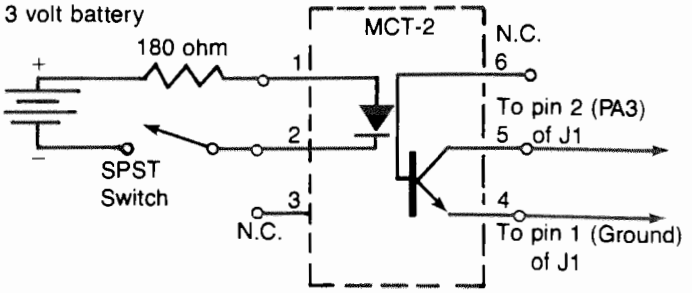


Figure 2 SAFETY ISOLATION

The 100 vocabulary Bliss Board is reprinted here through the courtesy of the Blissymbolics Communication Foundation.

	A	B	C	D	E	F	G	H	I	J
1	zero 0	one 1	two 2	three 3	four 4	five 5	six 6	seven 7	eight 8	nine 9
2	hello	question	I, me	(to) like	happy	action indicator	food	pen, pencil	friend	animal
3	goodbye	why	you	(to) want	angry	mouth	drink	paper, page	God	bird
4	please	how	man	(to) come	afraid	eye	bed	book	house	flower
5	thanks	who	woman	(to) give	funny	legs and feet	toilet	table	school	water, liquid
6	much, many	what thing	father	(to) make	good	hand	pain	television	hospital	sun
7	opposite meaning	which	mother	(to) help	big	ear	clothing	news	store	weather
8	music	where	brother	(to) think	young	nose	outing	word	showplace, theatre	day
9	print message	when	sister	(to) know	difficult	head	motor car	light	room	weekend
10		how much, many	teacher	(to) wash, bathe	hot	name	wheelchair	toy	street	birthday

```
100 REM*HELP
105 POKE 40963,0
110 S$="ABCDEFGHIJ"
115 INPUT"SCAN RATE";SR
120 FORX=1TO10
130 FORY=1TO10
140 READ D$(X,Y)
150 NEXTY:NEXTX
160 REM SCAN ROUTINE
170 FORX=1TO10
180 PRINTXTAB(5)L$
185 GOSUB2100
190 FORZ=1TOSR
200 P=PEEK(40961)
210 IFP<255THENGOSUB2000
220 IFP<255THEN250
225 NEXTZ
240 NEXTX
245 GOTO170
250 REM Y COORDINATE
260 FORY=1TO10
265 M$=L$
270 PRINTMID$(S$,Y,1)
275 GOSUB2100
280 Z=Z+1:P=PEEK(40961)
290 IFP<255THEN350
300 IFZ<SRTHEN280
310 Z=0
340 NEXTY
345 GOTO250
350 L$=D$(X,Y)
355 GOSUB2000
360 IFL$="---"THENPRINT!M$:L$=M$
362 IFL$="HELP"THENGOSUB2200
365 GOTO170
1000 DATA0,1,2,3,4,5,6,7,8,9
1005 DATAHELLO,?,I/ME,LIKE,HAPPY
1010 DATAACTION,FOOD,PENCIL
1012 DATAFRIEND,ANIMAL
1014 DATAGOODBYE,WHY,YOU,WANT
1015 DATAANGRY,MOUTH,DRINK,PAPER
1017 DATAGOD,BIRD,PLEASE,HOW,MAN
1020 DATACOME,AFRAID,EYE,BED,BOOK
1022 DATAHOUSE,FLOWERS,THANKS
1025 DATAWHO,WOMAN,GIVE,FUNNY,LEGS
1027 DATATOILET,TABLE
1030 DATASCHOOL,WATER,MUCH,WHAT,FATHER
1035 DATAMAKE,GOOD,HAND,PAIN
1037 DATATELEVISION,HOSPITAL,SUN
1040 DATAOPPOSITE,WHICH,MOTHER,HELP
1043 DATABIG,EAR,CLOTHING,NEWS
1045 DATASTORE,WEATHER,MUSIC,WHERE
1047 DATABROTHER,THINK
1050 DATAYOUNG,NOSE,OUTING,WORD
1053 DATATHEATER,DAY,----
1055 DATAWHEN,SISTER,KNOW,DIFFICULT
1058 DATAHEAD,CAR
1060 DATALIGHT,ROOM,WEEKEND,----
1063 DATAHOW MANY,TEACHER
1065 DATAWASH,HOT,NAME,WHEELCHAIR
1067 DATATOY,STREET,BIRTHDAY
2000 REM ALERT
2005 Z=0
2010 POKE40971,16
2020 POKE40970,15
2030 POKE40968,200
2040 FORU=1TO500:NEXT
2050 POKE40968,0
2060 RETURN
2100 REM BEEP
2110 POKE40971,16
2120 POKE40970,15
2125 FORV=1TO10
2130 POKE40968,100
2135 FORF=1TO20:NEXTF
2140 POKE40968,0
2150 NEXTV
2160 RETURN
2200 POKE40968,65
2205 PRINTTAB(5)"HELP"
2210 P=PEEK(40961)
2220 IFP=225THEN2210
2230 POKE40968,0
2240 RETURN
```

Program Remarks

1. Line 105: Sets the A data direction register to input mode.

2. Line 110: The SS string variable defines the horizontal coordinates and may be replaced by any set of 10 alphanumeric characters. For example, "0 1 2 3 4 5 6 7 8 9".

3. Lines 120–150: Reads and defines D$ string variables listed in data statements starting at line 1000.

4. Line 180: Displays the L$ string variable which holds English translation of selected Bliss symbol.

5. Lines 190–225: Looks at A side of the VIA. If the control switch is closed, pin 2 is brought low, the variable P is set to 251, the L$ string is defined by the current X, Y values and the "beep" subroutine is called.

6. Line 360: If the L$ string array is either 9,1 or 10,1 (i.e.,—) the M$ string is printed. The M$ string was set equal to the preceeding L$ string variable in line 265. The L$ string is also set equal to M$ to ensure that the current English translation is displayed.

A SHORT HISTORY OF BLISSYMBOLICS

Charles K. Bliss was intrigued by the way the Chinese people could communicate with each other across boundaries of dialect by using a set of standardized symbols. He wondered if someone could invent a language system that could surmount cultural barriers and be easily learned.

Bliss worked on such a language system while he lived in Australia and by 1949 was able to publish Semantography, the book that provides the explanation for his system of pictographs and ideographs. He intended that his symbols (known as Blissymbolics) be used as a universal language.

In 1971, a group at the Ontario Crippled Children's Centre started using Blissymbolics successfully with cerebral palsied, school age, non-speaking children. The Blissymbolics Communication Institute was then established as an international, non-profit service organization to maintain symbol standards and to provide training and materials for the people who apply Blissymbolics with non-speaking people.

For more information, contact BCI at 350 Rumsey Rd., Toronto, Ontario, Canada M4G 1R8 or call them at (416) 425-7835.

INDUSTRIAL SYSTEM SPOTLIGHT

EDITOR'S NOTE: The Industrial, and OEM (Original Equipment Manufacturers), uses for AIM 65 are many and varied. If you have developed a system around the AIM 65 that is used in an industrial or OEM application and would like it featured in INTERACTIVE, drop me a line with some of the details and a photo.

the Editor

Intended for use by power companies as a remote data acquisition system, the **MMS-9 MET Measurement System** by Dutec Inc. (4801 James McDivitt Rd., Jackson, MI 49204) uses an AIM 65 as its central processor.

The MMS-9 is equipped with sensor inputs to monitor meteorological and pollution data around the power generator and either store the recorded data (such as wind speed, direction, temperature, and sulphur dioxide content) or send it to another computer for processing. Twenty analog input channels are included in the basic unit.

According to John Dute, president of Dutec, the MMS-9 can, with the proper sensors, be used to measure wind dispersion around nuclear power plants. Mr. Dute goes on to mention that the major advantages of this system over the previous strip chart and magnetic recorder method of storing the measurements are cost, and having a real-time access to the data. "It's even possible for an agency like the Nuclear Regulatory Commission to have up-to-the-minute field measurements of every nuclear reactor installation as close as their phone," Mr. Dute added.

Mr. Dute further stated that previous "intelligent" solutions to solving this problem consisted of mini-computers which cost many times what the AIM 65 based system can sell for.

Here's an application where AIM 65 does the job cheaper and better than previous solutions.

Courtesy of Dutec Inc.

LINEAR PROGRAMMING IN BASIC

George J. Sellers
Cumberland, MD

Here is a Basic program you might find of interest for solving linear programming problems using the revised simplex method. This version will maximize the objective function with all constraints in the form < = a constant. The program dynamically allocates the arrays used and will solve problems of sizes up to those shown in the table for a 4K AIM 65.

ROW SIZE	COLUMN SIZE							
	2	3	4	5	6	7	8	9
2	X	X	X	X	X	X	X	X
3	X	X	X	X	X	X	X	
4	X	X	X	X	X			
5	X	X	X	X				
6	X	X	X					
7	X							
8								
9								

Representative run times for sample problems are shown in the following table:

ARRAY SIZE	RUN TIME
3 × 2	5.05 sec.
4 × 3	9.03 sec.
7 × 2	18.72 sec.
2 × 5	4.51 sec.
3 × 4	9.43 sec.

The input is organized with the coefficient of each constraint equation being a row of a matrix "A" which is called the coefficient matrix. The right side of the equations are organized into a column which is called the constant matrix "B". The solutions are also organized into a column which is called the solution matrix "X". The objective function is a row matrix "C".

Thus, the equations for each constraint are in the form A*X< =B in matrix algebra. The data are entered into the program by way of the prompts for each column. The coefficient matrix can then be printed out to verify the accuracy of the input and corrections made if necessary. Finally, the constant matrix can be input and the constants for the objective function are entered. In a short time the solution matrix is printed.

If you're not familiar with matrix notation, study the SCHAUM'S OUTLINE SERIES on LINEAR ALGEBRA by S. LIPSCHUTZ (McGraw-Hill, New York).

A good overview of many applications of linear programming in management as well as other areas on management science can be found in the book PRINCIPALS OF MANAGEMENT SCIENCE (2nd ed.) by H. M. Wagner (published by Prentice-Hall, NJ).

Linear programming forms the basis of many important types of problems that require optimization by maximizing or minimizing some function (this program solves only for maximums but, by using the procedures for inequalities, the input data can be rearranged so that minimization problems can also be solved).

```
RUN
ENTER IM & JM? 4
?? 3
COL  1 ROW  1 ? -2
COL  1 ROW  2 ? 1
COL  1 ROW  3 ? 2
COL  1 ROW  4 ? 1
COL  2 ROW  1 ? 3
COL  2 ROW  2 ? -2
COL  2 ROW  3 ? 1
COL  2 ROW  4 ? 1
COL  3 ROW  1 ? 0
COL  3 ROW  2 ? -4
COL  3 ROW  3 ? 1
COL  3 ROW  4 ? 5
CHECK INPUT? YES
ROW  1    -2    3    0
ROW  2     1   -2   -4
ROW  3     2    1    1
ROW  4     1    1    5
CHANGE INPUT? NO
B( 1 )? 2
B( 2 )? 5
B( 3 )? 6
B( 4 )? 10
C( 1 )? 2
C( 2 )? -3
C( 3 )? 3
MAX= 9.11111112
SOLUTION
  1  2.22222222
  2  0
  3  1.55555556
SLACK VARIABLES
  1  6.44444444
  2  9
  3  0
  4  0
```

```
0 REMLINEAR PROGRAMMING
5 INPUT"ENTER IM & JM";IM,JM:N=IM+JM
10 DIMA(IM,JM),B(IM),C(JM),X(JM),S(IM),AP(IM,N),CP(N),XP(N)
20 DIMBI(N,N),AR(N),Y(IM),LC(N),YY(IM),H(IM),LB(IM)
40 FORJ=1TOJM:FORI=1TOIM:PRINT"COL "J;"ROW ";I;
50 INPUTA(I,J):NEXTI:NEXTJ
```

```
60 INPUT"CHECK INPUT";AN$:IFAN$="NO"GOTO100
70 FORI=1TOIM:PRINT:PRINT"ROW ";I;"   ";
75 FORJ=1TOJM:PRINTA(I,J);
79 NEXTJ:NEXTI:PRINT
80 INPUT"CHANGE INPUT";AN$:IFAN$="NO"GOTO100
85 INPUT"ROW, COL, & NEW VALUE";I,J,A(I,J):GOTO80
100 FORI=1TOIM:PRINT"B(";I;")";
110 INPUTB(I):NEXTI
120 FORJ=1TOJM:PRINT"C(";J;")";
130 INPUTC(J):NEXTJ:GOSUB200
150 IFIB=1GOTO160
155 PRINT"UNBOUNDED":END
160 F=0:FORJ=1TOJM:F=F+C(J)*X(J):NEXTJ:PRINT"MAX=";F
170 PRINT"SOLUTION":FORJ=1TOJM:PRINTJ;X(J):NEXTJ
180 PRINT"SLACK VARIABLES":FORI=1TOIM:PRINTI;S(I):NEXTI
190 END
200 FORJ=1TOJM:LC(J)=0:CP(J)=C(J):XP(J)=0
210 FORI=1TOIM:AP(I,J)=A(I,J):NEXTI:NEXTJ
220 FORJ=1TOIM:LC(JM+J)=J:CP(JM+J)=0:XP(JM+J)=B(J):LB(J)=J+JM
230 FORI=1TOIM:IFI=JTHENBI(I,J)=1:GOTO260
250 BI(I,J)=0
260 AP(I,JM+J)=BI(I,J):NEXTI:NEXTJ
270 Z1=-.01:FORK=1TON:IFLC(K)<>0GOTO335
290 FORI=1TOIM:H(I)=0:FORJ=1TOIM:L=LB(J)
300 H(I)=H(I)+CP(L)*BI(J,I):NEXTJ:NEXTI
310 Z=0:FORI1TOIM:Z=Z+H(I)*AP(I,K):NEXTI
320 Z=Z-CP(K):IFZ<Z1THENIR=K:Z1=Z
335 NEXTK
340 IFZ1=-.01GOTO530
350 FORI=1TOIM:Y(I)=0:FORJ=1TOIM:Y(I)=Y(I)+BI(I,J)*AP(J,IR)
360 NEXTJ:NEXTI
370 T1=0.01:K=0:FORI=1TOIM:IFY(I)<.01GOTO430
380 L=LB(I):IFK=0THENIL=I:T1=XP(L)/Y(I)
400 IFXP(L)/Y(I)<T1THENIL=I:T1=XP(L)/Y(I)
420 K=1
430 NEXTI
440 IFT1=.01THENIB=0:RETURN
460 FORI=1TOIM:AR(I)=AP(I,IR):NEXTI:GOSUB600
480 FORI=1TOIM:IFI=ILGOTO500
490 L1=LB(I):L2=LB(IL):XP(L1)=XP(L1)-Y(I)/Y(IL)*XP(L2)
500 NEXTI
510 XP(IR)=XP(L2)/Y(IL):XP(L2)=0:LB(IL)=IR:LC(L2)=0
520 LC(IR)=IL:GOTO270
530 FORI=1TOJM:X(I)=XP(I):NEXTI
540 FORI=1TOIM:S(I)=XP(I+JM):NEXTI:IB=1:RETURN
600 IE=0:FORII=1TOIM:YY(II)=0:FORKK=1TOIM
605 YY(II)=YY(II)+BI(II,KK)*AR(KK):NEXTKK:NEXTII
620 IFABS(YY(IL))>=.000001GOTO630
625 RETURN
630 FORJJ=1TOIM:FORII=1TOIM:IFII=ILGOTO650
640 BI(II,JJ)=BI(II,JJ)-YY(II)/YY(IL)*BI(IL,JJ)
650 NEXTII
660 BI(IL,JJ)=BI(IL,JJ)/YY(IL):NEXTJJ:IE=1:RETURN
```

ASSEMBLY OFFSET

HOW TO MAKE THE AIM 65 ASSEMBLER OFFSET OBJECT CODE FOR EPROMS

Bruce McIntosh
National Research Council
Ottawa, Canada

Issue no. 2 of INTERACTIVE describes a modified tape loader program for offsetting object code which is to run in ROM-allocated memory. I have been using a method of writing source code which tricks the ASSEMBLER into doing the offsetting and bypasses tape storage and reloading. After the source code for a program has been written in this format, changing only one or two statements allows the following three options to be realized,

1) An assembled listing with all addresses in the listing correct as they will appear in ROM. No object code is produced.

2) Object code (with listing) which can be run and debugged in RAM.

3) Object code which is stored in RAM but is correct for running in ROM. This code is ready for dumping, usually via the TTY output, to an EPROM programmer. The listing from this assembly is not very useful.

My TV-monitor program runs in ROM beginning at $B3C0 and I shall use this as an example. At the beginning of the source code I define three constants.

i) The address where the program is to run, say

```
RUN  =  $B3C0
```

ii) The address where the object code is to be stored, either the final program or early versions which are to run and tested in RAM. In options 2) and 3) this will be a RAM address.

```
STORE  =  $0DC0
```

The last two hex digits do not have to be the same as the RUN address but this makes debugging easier.

iii) The difference between these two, which, to save EDITOR space, I usually designate by a single letter.

```
Z  =  RUN —STORE
```

After defining constants and absolute addresses for the program, the PROGRAM COUNTER is equated to STORE.

```
*  =  STORE
```

Then, in writing code to be processed by the ASSEMBLER, address labels which will be assigned by the ASSEMBLER are written with the shift factor "+Z" when they appear after a jump command. For example, if there is a subroutine labelled SCROLL, jumps to it are written

```
        JSR  SCROLL+Z
```

or,

```
        JMP  SCROLL+Z
```

Note that a branch is written normally

```
        BNE SCROLL
```

since the ASSEMBLER needs only the increment, not the actual address. The subroutine itself has its label written normally

```
SCROLL LDA CURSOR
        etc.
        etc.
```

Absolute addresses of course do not need the shift. For example, a MONITOR subroutine is defined by an equate and used normally;

```
SWSTAK  =  $EBBA
  • • • •
  • • • •
JSR SWSTAK
```

Admittedly, adding the "+Z" to the labels takes some thought and effort, but after the source code has been written in this form, the three options described at the beginning can be obtained by changing only the STORE and/or the RUN equate as illustrated in the following examples.

I wish to run and test the program in RAM. With the source code stored in the EDITOR I change the equates to

```
RUN  =  $0DC0
STORE  =  $0DC0
```

Obviously Z=0 and this is a normal program which is assembled and loaded for testing at $0DC0 in RAM.

When the program is completely debugged, I want a reference listing of the program as it will appear in ROM. I change

```
RUN  =  $B3C0
STORE  =  $B3C0
```

Here I run the ASSEMBLER with OBJ?=Y, OUT=X, and get a listing but no object code. Again Z=0 and it might appear that I am accomplishing very little. But now I make the changes

```
RUN   = $B3C0
STORE = $0DC0
```

When I assemble the program the resulting block of object code is loaded at $0DC0. It will not run there, but when it is transferred out to an EPROM it represents a program that will run correctly at $B3C0 in the B-ROM socket.

There are one or two tricky points. For example, the main entry point to the TV-monitor program is labelled OUTTV, and in the initialization the address assigned to OUTTV is loaded into the display linkage address (DILINK) of the main MONITOR by the following coding

```
LDA #<OUTTV
STA DILINK
LDA #>OUTTV
STA DILINK+1
```

Writing

```
LDA #<OUTTV+Z
```

produces an incorrect result. In this case it is necessary to do the address shift in an equate

```
OUTTVZ = OUTTV+Z
```

and write the source code as

```
LDA #<OUTTVZ
etc.
```

The subroutine is labelled in the normal manner

```
OUTTV PHA
  JSR PHXY
  etc.
```

The flexibility that this technique provides for EPROM program development is really quite surprising and will repay the added thought and effort required in writing the source code.

. . . COMING SOON

Forth for AIM 65

A new ROM set containing the FORTH programming language is expected to be available by the second quarter of 1981.

FORTH is a unique programming language that is well suited to a variety of applications. Because it was originally developed for real-time control systems, FORTH has features that make it ideal for machine and process control, energy management, data acquisition, automatic testing, robotics and other applications where assembly language was previously considered to be the only possible language choice.

AIM 65 FORTH is contained in two 4K byte ROMS which plug directly into the AIM 65 BASIC sockets. For further information, contact Electronic Devices Division, Rockwell International, POB 3669, RC55 Anaheim, CA 92803. The phone number is (714) 632-3729 or call your local Rockwell sales office.

HOW TO CHANGE THE STARTING ADDRESS FOR AIM 65 BASIC PROGRAMS

If you wish your BASIC programs to reside at a location other than the normal $0212 location, follow this simple procedure. First enter BASIC with the '5' key and answer all the prompts normally. Next, exit BASIC with the ESC key. If you'd like programs to start at $0500, modify the pointers at locations $0073 and $0075 to the following values:

```
0073 01 05
0075 03 05
```

and install three null bytes ($00) starting at 0500 0500 00 00 00

Now reenter BASIC with the '6' key and whatever programs are typed in or loaded from cassette will reside starting at $0500.

ZERO PAGE USAGE

In case you're wondering, here's a list of the zero page locations used by the AIM 65 system software.

AIM 65 MONITOR	$00AD-$00FE
ASSEMBLER ROM	$0004-$00AB
BASIC ROM	$0000-$00DB
PL-65 ROM	$0000-$0020, $0023

AIM 65 ASSEMBLER OUTPUT FORMATTER

. . . and Centronics printer driver

Georges-Emile April
Montreal, Quebec

(EDITOR'S NOTE: If you have a wide carriage printer hooked up to AIM 65, you're probably wishing that there were some way to reformat the output and make it more readable. Well, your wish is granted. And also a Centronics printer driver is thrown in to boot. PB0-PB7 is used for data output to the printer, CB2 is the 'data ready' line while CB1 is 'data received' line.)

```
;THE FOLLOWING OUTPUT DRIVER,ASSEMBLED HERE FOR USE WITH
;CENTRONICS 306C PRINTER,WILL REFORMAT ASSEMBLER OUTPUT FOR
;USE WITH LONG LINE PRINTERS
;IT REMOVES EXTRA CR/LF COMBINATIONS INSERTED BY THE AS-
;SEMBLER,AND ARRANGES LISTINGS IN NEAT COLLUMNS AS ONE CAN
;SEE IN THIS EXAMPLE.
;IT ALSO RECOGNISES TAB ($09) CHARACTERS,AND FILLS IN SPACES
;IT ALSO COUNTS LINES,AND GENERATES NEW PAGE AFTER SUFFI-
;CIENT NUMBER RECEPTION OF FORM-FEED CHARACTER ($C),CLEARS
;LINE COUNTER SO USER CAN CONTROL VERTICAL FORMAT IF HE
;WISHES
;THIS PROGRAM SHOULD BE ASSEMBLED AT ANY CONVENIENT ADDRESS


==0000                  *=$A479
==A479 LINCNT   EA      NOP
==A47A OLD      EA      NOP
==A47B CHRCNT   EA      NOP
==A47C FIRST    EA      NOP
==A47D NEW      EA      NOP
==A47E TEMP     EA      NOP
==A47F                  *=$E00
==0E00 START=*

==0E00 CRLF=$E9F0                       ==0E00 OUTFLG=$A413
==0E00                  *=$10A
               000E     .WOR USER


==010C DC1=$11
==010C DEL=$7F                          ==010C PLXY=$EBAC
==010C PHXY=$EB9E                       ==010C OUTALL=$E9BC
==010C WHEREQ=$E871                     ==010C OUTPUT=$EEFC
==010C UPB=$A000                        ==010C UDDRB=$A002
==010C UPCR=$A00C                       ==010C UIFR=$A00D
==010C UIER=$A00E
```

```
==010C                  *=START

==0E00 USER     B033    BCS EXEC
                ;THE FOLLOWING INITIALIZES DEVICE
                ;FOR OUTPUT TO OTHER TYPE OF DEVICE,
                ;THE FOLLOWING SHOULD BE CHANGED

==0E02 INIT     08      PHP
                A900    LDA #0
                8523    STA $23         ;CLEAR ASSEMBLER FLAG
                A90F    LDA #$F
                2D0CA0  AND UPCR
                09B0    ORA #$B0        ;PULSE ON CB2,POSITIVE
                8D0CA0  STA UPCR                EDGE ON CB1
                A914    LDA #$14
==0E13          8D0EA0  STA UIER
                8D0DA0  STA UIFR
                A900    LDA #00
                8D7BA4  STA CHRCNT      ;INIT COUNTER
                A90D    LDA #$D
                8D7AA4  STA OLD
==0E23          A9FF    LDA #$FF
                8D02A0  STA UDDRB
                A911    LDA #DC1
                8D00A0  STA UPB         ;INIT INTERFACE
                A90C    LDA #$C         ;GO TO NEW PAGE
                38      SEC
                20360E  JSR EXEC+1
==0E33          28      PLP
                60      RTS

==0E35 EXEC     68      PLA
                8D7DA4  STA NEW
                08      PHP
                209EEB  JSR PHXY
                F04E    BEQ IGNOR       ;IGNORE BLANKS
                C90A    CMP #$A         ;ELIMINATE LINEFEEDS
                F04A    BEQ IGNOR
                C97F    CMP #$7F
==0E45          F046    BEQ IGNOR       ;ELIMINATE DELETES
==0E47 FLUSH    AD7AA4  LDA OLD
                F028    BEQ COMMON
                C909    CMP #9
                D003    BNE *+5
                20C30E  JSR TABIT
                AE7BA4  LDX CHRCNT
                D03A    BNE NOTFRS
==0E58          8D7CA4  STA FIRST       ;MAKE NOTE OF FIRST
                C93D    CMP #'='                CHARACTER
                F00D    BEQ NOTAB
                C92A    CMP #'*
                F009    BEQ NOTAB
```

```
                A901   LDA #1
                2523   AND $23      ;TEST FOR PASS2
                F003   BEQ NOTAB    ;IF NOT,NO REFORMATTING
==0E69          20C30E JSR TABIT
==0E6C NOTAB    AD7AA4 LDA OLD
                F003   BEQ COMMON
                20E20E JSR OUTCHR
==0E74 COMMON   AD7DA4 LDA NEW
                8D7AA4 STA OLD
                A900   LDA #0
                8D7DA4 STA NEW
                AD7AA4 LDA OLD      ;RECOVER LAST CHARACTER
                A201   LDX #1
==0E84          E423   CPX $23      ;TEST FOR PASS2
                F005   BEQ IGNOR    ;IF SO LEAVE STACK AS IS
                AD7AA4 LDA OLD      ;IF NOT,FLUSH STACK
                D0BA   BNE FLUSH
==0E8D IGNOR    20ACEB JSR PLXY
                28     PLP
                60     RTS

==0E92 NOTFRS   C90D   CMP #$D      ;'CR' ?
                D0D6   BNE NOTAB
                AD7CA4 LDA FIRST
                C93D   CMP #'='
                D01D   BNE OTHER1
                2E7CA4 ROL FIRST
==0EA0 COM2     20C30E JSR TABIT
                AD7DA4 LDA NEW
                C93D   CMP #'=
                F0CA   BEQ COMMON
                8D7CA4 STA FIRST
                4C740E JMP COMMON
==0EB0 MAYBE    CD7CA4 CMP FIRST
                8D7CA4 STA FIRST
                F0B4   BEQ NOTAB
                D0E6   BNE COM2
==0EBA OTHER1   AD7DA4 LDA NEW
                C93B   CMP #';'
                F0EF   BEQ MAYBE
                D0A9   BNE NOTAB

==0EC3 TABIT    A920   LDA #' '
                20E20E JSR OUTCHR
                AD7BA4 LDA CHRCNT
                AA     TAX
                29E0   AND #$E0
                F004   BEQ LOW
                A907   LDA #7
                D002   BNE *+4
==0ED4 LOW      090F   ORA #$F
                2D7BA4 AND CHRCNT
                D0E8   BNE TABIT
                E020   CPX #$20
                F0E4   BEQ TABIT
                60     RTS

==0EE0 OUT2     A90A   LDA #$A      ;AFTER CR,INSERT LF
==0EE2 OUTCHR   201F0F JSR OUTIT
                C90D   CMP #$D      ;IS IT CR?
                F0F7   BEQ OUT2     ;IF SO ADD LF
                EE7BA4 INC CHRCNT   ;COUNT CHARACTER
                C90A   CMP #$A
                D002   BNE OUT3
                A90D   LDA #$D
==0EF2 OUT3     2C1E0F BIT BITS     ;TEST FOR NON-PRINTING
                D003   BNE OUT4              CHARACTERS
                CE7BA4 DEC CHRCNT   ;NON-PRINTING CHARACTERS
==0EFA OUT4     A200   LDX #0            MUST NOT BE COUNTED
                C90D   CMP #$D      ;TEST FOR CR
                D017   BNE NOCR
                8E7BA4 STX CHRCNT   ;IN CASE OF CR,CLEAR
                                      CHARACTER COUNTER

                EE79A4 INC LINCNT   ;AND INCREMENT LINE
                A03C   LDY #60               COUNTER
                CC79A4 CPY LINCNT   ;TOO MANY LINES?
==0F0B          D00A   BNE NOCR
                A90C   LDA #$C
                201F0F JSR OUTIT
                A90D   LDA #$D
                8E79A4 STX LINCNT
==0F17 NOCR     C90C   CMP #$C
                D003   BNE BITS
                8E79A4 STX LINCNT
==0F1E BITS     60     RTS


;THE FOLLOWING ROUTINE DOES THE ACTUAL OUTPUT TO PRINTING
;DEVICE. IT SHOULD BE CHANGED IF OUTPUT TO DIFFERENT DEVICE
;IS DESIRED

==0F1F OUTIT    48     PHA          ;SAVE CHARACTER
                A910   LDA #$10
==0F22 WAIT     2C0DA0 BIT UIFR
                F0FB   BEQ WAIT     ;WAIT FOR PRINTER READY
                8D0DA0 STA UIFR     ;CANCEL 'READY FLAG'
                68     PLA          ;GET CHARACTER
                8D00A0 STA UPB      ;SEND TO PRINTER
                60     RTS
                       END D306C    ;POINT TO COMMENT FILE
                ERRORS= 0000
```

AIM 65 TAPE CATALOG PROGRAM

Steve Bresson
Severn, Md.

(EDITOR'S NOTE: How many times have you forgotten which programs were on a certain cassette tape? It's happened to me more than once. Here's a program that will not only tell you the names of the programs on tape, but will also tell you what type of program it is.)

When programs are saved on tape by the AIM 65 there are normally 2 distinct formats. In my basic data subroutines I add another format so that the program will know that it is reading a data file:

1) Object — begins with a <CR>.
2) Text — begins with a <space>.

   a) Text — ends with a <CR><CR>.
   b) Basic program — ends with a <CTL-Z>.

3) Basic Data — begins with a <#><CR>.
   Ends with a <CTL-Z>.
   This is a TEXT file.

TLIST reads the tape looking for the beginning of any file. When found, it lists the filename, and a T, O, or D to indicate a Text, Object, or Data file. For text and data files it then reads the file, looking for a <CTL-Z> or double <CR> to end the file. If a <CTL Z> is found, a B is put out to indicate a Basic file. The last item on the line is the length of the file in hex. For object files the program lists the starting and ending addresses of each continuous block in the file.

The only way to get out of the program, after you have listed all your files, is to do a reset---the tape input program is too busy to bother with scanning the keyboard and ignores you.

SUBROUTINES

LIST	022B	Used here to print a prompt at the start of the program
GET	0240	Gets input from the keyboard and echos it. Ends on <CR>.
CR	02F2	Puts out a <CR> and clears the character counter (OCNTR).
FINDF	02FA	Find a file. Reads from tape to find the beginning of a file. Prints out the file name and O/T/D. Returns with CY set if T or D type file. CY clear for O type file.
OBJ	027C	Searches for continuous blocks in an object file and lists the start/end addresses. Ends on a 0 length record.
TEXT	02CC	Parses through a text or data file looking for a <CTL Z> or <CR><CR>. Prints out file length in hex.
TAB	021A	Puts out spaces to align the output fields.
ENDBLK	024B	Prints out the end address of an object block.
PRNTYX	025D	Tabs over to the correct column and prints the contents of TMPY and TMPX. Changes OLD to the new start address.
UPDATE	0298	Updates the address variable so we know if the next record is continuous. If the address it trys to use is not correct, it calls ENDBLK and starts a new line for the new block.
GBYT	02C2	Reads a byte from the tape input subroutine and bumps the counter.
FBLKST	033D	Parses thru the tape input looking for the start of a record (object format).
BNK	0355	Bumps the output character counter then jumps to BLANK to print a space.
OUTP	035A	Bumps the output character counter then jumps to OUTPUT to print to the display/ printer.

If you start the program up (at $0200) by using the F1 key (it must be initialized), you will see:

<[>TAPE=^

This is a prompt for you to put in any information you wish to have printed. I usually put in the tape i.d. and the date. The program will then turn on the tape and begin looking for a file. If the monitor subroutine becomes confused because of garbage on the tape, it will print 'ERROR' and jump to the monitor. This will not hurt anything. Just start the program up again and continue from where you are.

EXAMPLE:

```
<[>TAPE=TEST TAPE
DATEX   TB   042F                  —basic source file
TESTO   DB   005E                  —basic data file
JUNK    T    030F                  —text file
KCMD    O    00CA   00DE          —object file with 2 segments
             010C   010E
SUBMN   O    A400   A401          —object file with 3 segments
             010F   0111
             0200   03C9
MEMO1   O    0000   0114          —object file
```

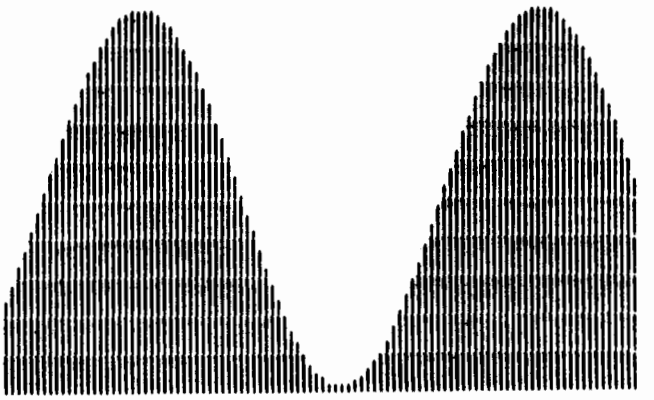
```
2000                    BLANK   =$E83E
2000                    RDRUB   =$E95F
2000                    BLANK2  =$E83B
2000                    WRAX    =$EA42
2000                    TIBYTE  =$ED3B
2000                    OUTPUT  =$E97A
2000                    CRLF    =$E9F0
2000                    RCHEK   =$E907
2000                    BLK     =$0115
2000                    TIBY1   =$ED53
2000                    PRIFLG  =$A411
2000                    CURPO2  =$A415
2000                    TABUFF  =$0116
2000                    CTLZ    =$1A
2000
2000                            *=$0000
0000                    OLD     *=*+2
0002                    TMPA    *=*+1
0003                    TMPX    *=*+1
0004                    TMPY    *=*+1
0005                    OBJFLG  *=*+1
0006                    TPCTR   *=*+2
0008                    OCNTR   *=*+1
0009                    TCFLG   *=*+1
000A
000A                            *=$0200
0200
0200
0200                    ;TURN PRINTER ON
0200  A9 80                       LDA #$80
0202  8D 11 A4                    STA PRIFLG
0205  20 30 02                    JSR LIST
0208  20 45 02                    JSR GET
020B  20 F7 02                    JSR CR
020E  20 FF 02            MAIN    JSR FINDF
0211  B0 06                       BCS M01
0213  20 81 02                    JSR OBJ
0216  4C 0E 02                    JMP MAIN
0219  20 D1 02            M01     JSR TEXT
021C  4C 0E 02                    JMP MAIN
021F  48                  TAB     PHA
0220  A5 08                TAB1    LDA OCNTR
0222  C9 09                       CMP #09
0224  B0 08                       BCS TAB2
0226  20 3E E8                    JSR BLANK
0229  E6 08                       INC OCNTR
022B  4C 20 02                    JMP TAB1
022E  68                  TAB2    PLA
022F  60                          RTS
0230                      LIST
0230  A2 00                       LDX #00
0232  BD 3F 02            LIST1   LDA LDAT,X
0235  F0 07                       BEQ LIST2
0237  20 7A E9                    JSR OUTPUT
023A  E8                          INX
023B  4C 32 02                    JMP LIST1
023E  60                  LIST2   RTS
023F  54 41   LDAT        .BYTE 'TAPE=',00
0244  00
0245  A0 00                GET     LDY #00
0247  20 5F E9            GET1    JSR RDRUB
024A  C8                          INY
024B  C9 0D                       CMP #$0D
024D  D0 F8                       BNE GET1
024F  60                          RTS
0250
0250  20 57 03            ENDBLK  JSR BNK2
0253  A6 00               ENDBK1  LDX OLD
0255  A5 01                       LDA OLD+1
0257  CA                          DEX
0258  E0 FF                       CPX #$FF
025A  D0 03                       BNE EN2
025C  38                          SEC
025D  E9 01                       SBC #01
025F  4C 7A 02            EN2     JMP PRAX
0262  20 1F 02            PRNTYX  JSR TAB
0265  A5 04                       LDA TMPY
0267  A6 03                       LDX TMPX
0269  20 7A 02                    JSR PRAX
026C  A5 02                       LDA TMPA
026E  18                          CLC
026F  65 03                       ADC TMPX
0271  85 00                       STA OLD
0273
0273  A9 00                       LDA #00
0275  65 04                       ADC TMPY
0277  85 01                       STA OLD+1
0279  60                          RTS
027A
027A  E6 08                PRAX    INC OCNTR
027C  E6 08                        INC OCNTR
027E  4C 42 EA                     JMP WRAX
0281  A9 80                OBJ     LDA #$80
0283  85 05                        STA OBJFLG
0285  20 42 03             OBJ1    JSR FBLKST
0288  D0 06                        BNE OBJ1A
028A  20 50 02                     JSR ENDBLK
028D  4C F7 02                     JMP CR
0290  20 9D 02             OBJ1A   JSR UPDATE
0293  20 3B ED             OBJ2    JSR TIBYTE
0296  C6 02                        DEC TMPA
0298  D0 F9                        BNE OBJ2
029A  4C 85 02                     JMP OBJ1
029D  85 02                UPDATE  STA TMPA
029F  86 03                        STX TMPX
```

```
02A1  84 04           STY TMPY
02A3  24 05           BIT OBJFLG
02A5  10 05           BPL UP1
02A7  46 05           LSR OBJFLG
02A9  4C 62 02        JMP PRNTYX
02AC  C4 01     UP1   CPY OLD+1
02AE  F0 09           BEQ UP3
02B0  20 50 02  UP2   JSR ENDBLK
02B3  20 F7 02        JSR CR
02B6  4C 62 02        JMP PRNTYX
02B9  E4 00     UP3   CPX OLD
02BB  D0 F3           BNE UP2
02BD  18              CLC
02BE  65 00           ADC OLD
02C0  85 00           STA OLD
02C2  90 02           BCC UP4
02C4  E6 01           INC OLD+1
02C6  60        UP4   RTS
02C7  20 3B ED  GBYT  JSR TIBYTE
02CA  E6 00           INC OLD
02CC  D0 02           BNE GBY1
02CE  E6 01           INC OLD+1
02D0  60        GBY1  RTS
02D1  A9 00     TEXT  LDA #00
02D3  85 00           STA OLD
02D5  85 01           STA OLD+1
02D7  20 C7 02  TX1   JSR GBYT
02DA  C9 1A     TX2   CMP #CTLZ
02DC  D0 08           BNE TX3
02DE  A9 42           LDA #'B
02E0  20 5F 03        JSR OUTP
02E3  4C F1 02        JMP TX4
02E6  C9 0D     TX3   CMP #$0D
02E8  D0 ED           BNE TX1
02EA  20 C7 02        JSR GBYT
02ED  C9 0D           CMP #$0D
02EF  D0 E9           BNE TX2
02F1  20 1F 02  TX4   JSR TAB
02F4  20 53 02        JSR ENDBLK+3
02F7  20 F0 E9  CR    JSR CRLF
02FA  A9 00     CR1   LDA #00
02FC  85 08           STA OCNTR
02FE  60              RTS
02FF  20 07 E9  FINDF JSR RCHEK
0302  A2 00           LDX #00
0304  A9 00           LDA #00
0306  8D 15 01        STA BLK
0309  20 53 ED        JSR TIBY1
030C  CA              DEX
030D  8E 15 A4        STX CURP02
0310  AD 16 01        LDA TABUFF
0313  C9 00           CMP #00
0315  D0 E8           BNE FINDF
0317  A2 05           LDX #$05
0319  20 3B ED  FND1  JSR TIBYTE
031C  20 5F 03        JSR OUTP
031F  CA              DEX
0320  D0 F7           BNE FND1
0322  20 5A 03        JSR BNK
0325  20 3B ED        JSR TIBYTE
0328  C9 0D           CMP #$0D
032A  D0 07           BNE FND2
032C  A9 4F           LDA #'O
032E  20 5F 03        JSR OUTP
0331  18              CLC
0332  60              RTS
0333
0333  C9 23     FND2  CMP #'#
0335  D0 04           BNE FND4
0337  A9 44           LDA #'D
0339  D0 02           BNE FND5
033B  A9 54     FND4  LDA #'T
033D  20 5F 03  FND5  JSR OUTP
0340  38              SEC
0341  60              RTS
0342  20 3B ED  FBLKST JSR TIBYTE
0345  C9 3B           CMP #'
0347  D0 F9           BNE FBLKST
0349  20 3B ED        JSR TIBYTE
034C  48              PHA
034D  20 3B ED        JSR TIBYTE
0350  A8              TAY
0351  20 3B ED        JSR TIBYTE
0354  AA              TAX
0355  68              PLA
0356  60              RTS
0357
0357  20 5A 03  BNK2  JSR BNK
035A  E6 08     BNK   INC OCNTR
035C  4C 3E E8        JMP BLANK
035F  E6 08     OUTP  INC OCNTR
0361  4C 7A E9        JMP OUTPUT
0364                  .END
```

SOLID GRAPH PLOT

Mike Corder
Jim Nickum
Rick Ketchum

(EDITOR'S NOTE: The following machine code is a modified version of the AIM PLOT routine (published in issue #2) that does a solid graph instead of the dots. It's very striking as you can see. The BASIC program was written by Tex Thomas of Rockwell to plot a sine wave. You can experiment with different functions.)



Here's an example of a sine wave plotted by the Solid Graph Plot routine.

```
2000                        *=$0EF0
0EF0
0EF0                PAT23   =$FFA0
0EF0                PRIERR  =$F079          ;MONITOR SR
0EF0                PCR     =$A80C          ;PRINTER CONTROL REG
0EF0                PRST    =0              ;PRINTER START (CB1)
0EF0                SP12    =1              ;STROBE P1,P2 (CA1)
0EF0                MON     =$C0            ;MOTOR ON (CB2=0)
0EF0  A9 C1         MTRON   LDA #PRST+SP12+MON
0EF2  8D 0C A8              STA PCR
0EF5  20 A0 FF              JSR PAT23       ;CHECK FOR RUNNING
0EF8  D0 08                 BNE CONT
0EFA  20 A0 FF              JSR PAT23       ;AGAIN
0EFD  D0 03                 BNE CONT
0EFF  4C 79 F0              JMP PRIERR      ;MOTOR FAIL MSG
0F02  60            CONT    RTS
0F03                IOUTU   =$A479          ;TOP TWO
0F03                IOUTL   =$A478          ;BOTTOM 8 ELEMENTS
0F03                DRB     =$A800
0F03                DRAH    =               DRB+1 ;DATA REG A
0F03                T2L     =$A808          ;TIMER 2 LATCH - LOW
0F03                T2H     =               T2L+1 ;TIMER 2 LATCH-HIGH
0F03                DE2     =$EC1B          ;TIMER DELAY ROUTINE
0F03                PRTIME  =$1110          ;DOT PRINT TIME (MS)
0F03                ;
0F03  AD 79 A4      PRDOT   LDA IOUTU       ;UPPER MASK
0F06  0D 00 A8              ORA DRB         ;WITH PRESENT
0F09  8D 00 A8              STA DRB
0F0C  AD 78 A4              LDA IOUTL       ;LOWER
0F0F  8D 01 A8              STA DRAH        ;TURN THEM ON
0F12  A9 10                 LDA #<PRTIME
0F14  8D 08 A8              STA T2L         ;SET TIMER
0F17  A9 11                 LDA #>PRTIME
0F19  8D 09 A8              STA T2H
0F1C  20 1B EC              JSR DE2         ;DELAY
0F1F  A9 00                 LDA #0
0F21  8D 01 A8              STA DRAH        ;TURN IT OFF
0F24  AD 00 A8              LDA DRB
0F27  29 FC                 AND #$FC        ;MASK OFF ELEMENTS
```

```
0F29  8D 00 A8          STA DRB
0F2C  60                RTS
0F2D                ;
0F2D                DATA    =$E12           ;BEGINNING OF PLOT DATA
0F2D  A9 00         CALC    LDA #0
0F2F  8D 78 A4          STA IOUTL
0F32  8D 79 A4          STA IOUTU
0F35  B9 12 0E          LDA DATA,Y          ;NEXT PLOT POINT
0F38  20 CD 0F          JSR CNVT            ;CONVERT TO OUR FORMAT
0F3B  48                PHA
0F3C  29 0F             AND #$0F            ;GET DOTNUM
0F3E  8D F6 0F          STA DOTNUM
0F41  8D F4 0F          STA SADOT           ;SAVE FOR COUNT
0F44  68                PLA
0F45  C8                INY
0F46  29 F0             AND #$F0            ;MASK ELEM NUMBER
0F48  4A                LSR A
0F49  4A                LSR A
0F4A  4A                LSR A
0F4B  4A                LSR A
0F4C  C9 08             CMP #8
0F4E  90 11             BCC SHIFT           ;BRANCH IF <8
0F50  D0 04             BNE CONT1           ;BRANCH IF =9
0F52  A9 01             LDA #1              ;SET 8 MASK
0F54  D0 02             BNE PAT4
0F56  A9 03         CONT1   LDA #3          ;9
0F58  8D 79 A4      PAT4    STA IOUTU
0F5B  A9 FF             LDA #$FF            ;TURN LOW ALL ON
0F5D  8D 78 A4          STA IOUTL
0F60  60            END     RTS
0F61                ;
0F61  AA            SHIFT   TAX
0F62  38                SEC
0F63  2E 78 A4      LOOP1   ROL IOUTL           ;CARRY INTO MASK
0F66  E0 00             CPX #0              ;DONE?
0F68  F0 F6             BEQ END
0F6A  38                SEC
0F6B  CA                DEX
0F6C  4C 63 0F          JMP LOOP1
0F6F                ;
0F6F                ACR     =$A80B          ;AUX CONTROL REG
0F6F                DTSP    =4850           ;DOT DELAY TIME
0F6F                STTIME  =4500           ;START DELAY TIME
0F6F                ;
0F6F  A0 01         MAIN    LDY #1
0F71  20 2D 0F      LOOP2   JSR CALC            ;EXTRACT PLOT POINT
0F74  20 F0 0E          JSR MTRON           ;MOTOR ON
0F77  A9 00             LDA #0
0F79  8D 0B A8          STA ACR
0F7C  A9 94             LDA #<STTIME
0F7E  8D 08 A8          STA T2L             ;DELAY TIME
0F81  A9 11             LDA #>STTIME
0F83  8D 09 A8          STA T2H
0F86  20 1B EC          JSR DE2             ;WAIT
```

```
0F89  AD F6 0F     LOOP3  LDA DOTNUM        ;SEE IF AT DOT POSITION
0F8C  D0 30               BNE CONT2         ;BRANCH IF > ZERO
0F8E                ; HAVE NOW PRINTED DOTS IN ALL THE RIGHT
0F8E                ; POSITIONS UP TO THE DOT POSITION ITSELF.
0F8E                ; FILL OUT COLUMNS BELOW UP TO START
0F8E                ; OF ADJACENT HEAD
0F8E  20 03 0F            JSR PRDOT         ;PRINT IT
0F91  18                  CLC
0F92  6E 79 A4            ROR IOUTU         ;PRINT LESS THAN POINT
0F95  B0 06               BCS Z4            ;WERE IN 8 OR 9 COLUMN
0F97  6E 78 A4            ROR IOUTL
0F9A  4C A2 0F            JMP Z2
0F9D  A9 FF        Z4     LDA #$FF          ;ALL LOW ON STILL
0F9F  8D 78 A4            STA IOUTL
0FA2  18           Z2     CLC
0FA3  AD F4 0F            LDA SADOT         ;SEE IF WE'RE THERE
0FA6  69 01               ADC #1
0FA8  8D F4 0F            STA SADOT
0FAB  C9 0A               CMP #10           ;DONE?
0FAD  F0 06               BEQ Z1
0FAF  20 03 0F            JSR PRDOT         ;KEEP PRINTING
0FB2  4C A2 0F            JMP Z2
0FB5  CE F5 0F     Z1     DEC NUM
0FB8  D0 B7               BNE LOOP2         ;MORE TO PLOT
0FBA  20 C7 0F            JSR MTROFF        ;TURN IT OFF
0FBD  60                  RTS
0FBE                ; PRINT UNTIL AT CORRECT DOT
0FBE  20 03 0F     CONT2  JSR PRDOT
0FC1  CE F6 0F            DEC DOTNUM
0FC4  4C 89 0F            JMP LOOP3
0FC7                ;
0FC7                ; MOTOR OFF
0FC7                ;
0FC7                MOFF   =$E0              ;CB2=1
0FC7  A9 E1         MTROFF LDA #PRST+SP12+MOFF
0FC9  8D 0C A8             STA PCR
0FCC  60                   RTS
0FCD                ;
0FCD                ; HEX CONVERT
0FCD                ;
0FCD  8D F3 0F     CNVT   STA HEX1
0FD0  98                  TYA
0FD1  48                  PHA
0FD2  F8                  SED
0FD3  A0 00               LDY #0
0FD5  A9 00               LDA #0
0FD7  18           LOOP4  CLC
0FD8  4E F3 0F            LSR HEX1
0FDB  90 04               BCC CONT3
0FDD  18                  CLC
0FDE  79 EC 0F            ADC VALUE,Y
0FE1  C8           CONT3  INY
0FE2  C0 07               CPY #7
```

```
0FE4  D0 F1              BNE LOOP4
0FE6  D8                 CLD                ;BACK TO BINARY
0FE7  AA                 TAX                ;SAVE CONVERTED DATA
0FE8  68                 PLA
0FE9  A8                 TAY
0FEA  8A                 TXA                ;DATA INTO A
0FEB  60                 RTS
0FEC  01          VALUE  .BYT 1,2,4,8,$16,$32,$64  ;DECIMAL
0FED  02
0FEE  04
0FEF  08
0FF0  16
0FF1  32
0FF2  64
0FF3  00          HEX1   .BYT 0
0FF4              SADOT  *=*+1
0FF5              NUM    *=*+1               ;COUNT OF POINTS (MAX=255)
0FF6              DOTNUM *=*+1               ;WHICH DOT
0FF7                     .END
```

```
10 REM INITIALIZE
11 :
20 INPUT"PLOT LENGTH";L
25 IF L>255 THEN L=255
30 DIM Y(L)
40 POKE 4085,L           :REM L TO $0FF5
50 POKE 4,111:POKE 5,15 :REM USR=$0F6F
90 :
99 :
100 REM  GENERATE DATA IN Y(L)
101 REM  &MAX, MIN OF Y(L)
102 :
110 M=100:C=100:D=7.9
120 FORX=1 TO L
130 : Y(X)=INT(C+M*SIN(X/D))
140 : IF Y(X)>MAX THEN MAX=Y(X)
150 : IF Y(X)<MIN THEN MIN=Y(X)
160 NEXT
170 :
180 :
190 :
500 REM  SCALE DATA 0-99
505 :
510 DS=3568              :REM DATA - $0DEF
520 AMP=MAX-MIN
530 SCALE=99/AMP
535 :
540 FOR X=1 TO L
550 : D%=(Y(X)-MIN)*SCALE + .5
560 : POKE DS+X,D%
570 NEXT
580 :
590 :
700 REM PLOT DATA
710 GOSUB 810
720 D=USR(D)
730 GOSUB 810
740 :
750 :
799 END
800 REM  4 LF
810 FOR X=1 TO 4
820 : PRINT! " "
830 NEXT
840 RETURN
```

IMPROVED PLOT ROUTINE

Marvin D. Shafer
Provo, Utah

(EDITOR'S NOTE: A number of you mentioned having problems with the original plot routine published in issue #2 of Interactive. For your benefit, here's an improved plot routine that works! (I tried it myself). Also, it can be interfaced to BASIC a bit more easily.)

This plot routine accepts and plots values in the accumulator ranging from 0 to 137. The routine determines which points it can plot and leaves a blank for those positions that are impossible to access like 5 and 6, 12 and 13 and so on (see the test result). The result is a linear plot.

TEST PROGRAM: (run this to see the linear plot example after PLOT has been entered)

```
0200 LDA #00
0202 STA 00
0204 JSR OEF5
0207 INC 00
0209 LDA 00
020B CMP #8A
020D BNE 0204
020F BRK
```

This new PLOT is also set up so that it can be called from BASIC with the USR function which has been initialized to point to $0EF0.

Example of a BASIC routine which calls PLOT:

```
10 POKE 4,240:POKE 5,14
15 Y=137/2
20 FOR X=0 TO 1600 STEP 4
26 Z=Y+Y*SIN (X/57.3)
35 W=USR (Z)
37 NEXT X
40 STOP
```

```
2000                    PRIFLA  =$A411
2000                    PINT    =$F0CB
2000                    PCR     =$A80C
2000                    PAT23   =$FFA0
2000                    PRIERR  =$F079
2000                    IDOT    =$A477
2000                    DRAH    =$A801
2000                    IFR     =$A80D
2000                    IOUTU   =$A479
2000                    DRB     =$A800
2000                    IOUTL   =$A478
2000                    T2L     =$A808
2000                    T2H     =$A809
2000                    INCP    =$F121
2000                    IBUFM   =$A460
2000                    IBITL   =$A47A
2000                    IBITU   =$A47B
2000                    NUM     =$FF
2000                    OUTPR   =$F038
2000
2000                            *=$0EF0
0EF0
0EF0  20 FE BE                  JSR $BEFE
0EF3  A5 AD                     LDA $AD
0EF5                    LABLE
0EF5  20 92 0F                  JSR VALDOT
0EF8  20 FC 0E                  JSR ALGRA
0EFB  60                        RTS
0EFC  2C 11 A4          ALGRA   BIT PRIFLA
0EFF  10 2A                     BPL OUT
0F01  20 CB F0                  JSR PINT
0F04  20 62 0F                  JSR NIPSU
0F07  A9 C1                     LDA #$C1
0F09  8D 0C A8                  STA PCR
0F0C  20 A0 FF                  JSR PAT23
0F0F  D0 08                     BNE NIP02
0F11  20 A0 FF                  JSR PAT23
0F14  D0 03                     BNE NIP02
0F16  4C 79 F0                  JMP PRIERR
0F19  20 2C 0F          NIP02   JSR NPDOT
0F1C  20 2C 0F                  JSR NPDOT
0F1F  AD 77 A4                  LDA IDOT
0F22  C9 0A                     CMP #$0A
0F24  90 F3                     BCC NIP02
0F26  A9 E1                     LDA #$E1
0F28  8D 0C A8                  STA PCR
0F2B  60                OUT     RTS
0F2C  A9 00             NPDOT   LDA #$00
0F2E  8D 01 A8                  STA DRAH
0F31  AD 0D A8          NDOTO   LDA IFR
0F34  29 02                     AND #$02
0F36  F0 F9                     BEQ NDOTO
0F38  AD 0C A8                  LDA PCR
0F3B  49 01                     EOR #$01
0F3D  8D 0C A8                  STA PCR
0F40  EE 77 A4                  INC IDOT
0F43  AD 79 A4                  LDA IOUTU
0F46  0D 00 A8                  ORA DRB
0F49  8D 00 A8                  STA DRB
0F4C  AD 78 A4                  LDA IOUTL
0F4F  8D 01 A8                  STA DRAH
0F52  A9 A4                     LDA #$A4
0F54  8D 08 A8                  STA T2L
0F57  A9 06                     LDA #$06
0F59  8D 09 A8                  STA T2H
0F5C  20 62 0F                  JSR NIPSU
0F5F  4C BA F0                  JMP $F0BA
0F62  A2 00             NIPSU   LDX #$00
0F64  20 21 F1                  JSR INCP
0F67  BD 60 A4          NIPS1   LDA IBUFM,X
```

```
0F6A  CD 77 A4          CMP IDOT
0F6D  D0 16             BNE NIPS3
0F6F  AD 7A A4          LDA IBITL
0F72  F0 08             BEQ NIPS2
0F74  0D 78 A4          ORA IOUTL
0F77  8D 78 A4          STA IOUTL
0F7A  D0 09             BNE NIPS3
0F7C  AD 7B A4  NIPS2   LDA IBITU
0F7F  0D 79 A4          ORA IOUTU
0F82  8D 79 A4          STA IOUTU
0F85  0E 7A A4  NIPS3   ASL IBITL
0F88  2E 7B A4          ROL IBITU
0F8B  CA                DEX
0F8C  CA                DEX
0F8D  10 D8             BPL NIPS1
0F8F  4C 18 F1          JMP $F118
0F92            VALDOT
0F92  85 FF             STA NUM
0F94  A0 00             LDY #00
0F96  A2 04             LDX #04
0F98            HERE
0F98  E4 FF             CPX NUM
0F9A  B0 1C             BCS SPRINT
0F9C  E8                INX
0F9D  E4 FF             CPX NUM
0F9F  F0 12             BEQ C
0FA1  E8                INX
0FA2  E4 FF             CPX NUM
0FA4  F0 0D             BEQ C
0FA6  CA                DEX
0FA7  CA                DEX
0FA8            ;ADD 7 TO X
0FA8  8A                TXA
0FA9  18                CLC
0FAA  69 07             ADC #07
0FAC  AA                TAX
0FAD            ;ADD 2 TO Y
0FAD  C8                INY
0FAE  C8                INY
0FAF  E0 90             CPX #144
0FB1  D0 E5             BNE HERE
0FB3            C
0FB3  A9 64             LDA #100
0FB5  4C BF 0F          JMP PRINT
0FB8            SPRINT
0FB8  A5 FF             LDA NUM
0FBA  84 FF             STY NUM
0FBC  38                SEC
0FBD  E5 FF             SBC NUM
0FBF            PRINT
0FBF  48                PHA
0FC0  A2 00             LDX #$00
0FC2  20 38 F0          JSR OUTPR
0FC5  A2 00             LDX #$00
```

TAPE PROBLEMS

Mark Reardon
Rockwell International

In recent months, it has come to our attention that a lot of AIM 65 users have been experiencing cassette tape problems. Most of these result from their choice in cassette recorders and/or tapes.

We have found the most successfully used decks are the General Electric models 3-5XXX. There are several different styles available with different options to satisfy most needs.

Using Chromium Dioxide or Metal tapes with conventional recorders causes the high frequency end to be muted. Reading errors occur since this high end is what is needed most by the AIM 65.

The solution is to pick a good quality tape of the appropriate type of bias for the users' deck. This will not only save him from frustration but also it will usually save him a few dollars for cassettes.

```
0FC7  68                PLA
0FC8  C9 05     DIVA    CMP #$05
0FCA  90 05             BCC FEIN
0FCC  E9 05             SBC #$05
0FCE  E8                INX
0FCF  D0 F7             BNE DIVA
0FD1  18        FEIN    CLC
0FD2  2C 82 EF          BIT $EF82
0FD5  08                PHP
0FD6  49 03             EOR #$03
0FD8  69 01             ADC #$01
0FDA  28                PLP
0FDB  F0 02             BEQ SPEI
0FDD  29 03             AND #$03
0FDF  9D 60 A4  SPEI    STA IBUFM,X
0FE2  8A                TXA
0FE3  2C 97 F0          BIT $F097
0FE6  D0 08             BNE ZUR
0FE8  BD 60 A4          LDA IBUFM,X
0FEB  69 05             ADC #$05
0FED  9D 60 A4          STA IBUFM,X
0FF0  60        ZUR     RTS
0FF1
0FF1                    .END
```

LETTERS TO THE EDITOR

Dear Editor,

Many thanks for an excellent newsletter. I was particularly interested in Ken Fullbrook's letter in issue #3 as I have also been using the EDITOR directly for BASIC. A simple routine I use avoids having to initialize the EDITOR using <T> and does not need a SPACE at the beginning of the top line. This INPUT HANDLER is as follows:

```
UIN       .WORD INTST     e.g. Ø1Ø8 = ØØ , Ø1Ø9 =
                              ØF
INTST     BCC IPINIT      ØFØØ BCC ØFØ5
          JMP MREAD       ØFØ2 JMP FADØ
IPINIT    JMP TOPNO       ØFØ5 JMP F8BC
```

To LOAD the program to BASIC answer "U" to the IN prompt just as Ken describes, and *remember* the bottom line of the EDITOR must be CTRL/Z.

A tip now for those users with cassette tape recorders which do not have a tape counter. The VERIFY command <3> is useful for scanning tapes containing more than one DUMPED or LISTED file, and scanning with <3> for a particular file terminates the scan at the end of this file. This is useful for finding where on the tape the next file can be recorded. This is fine until files are SAVED from BASIC when <3> never terminates. This occurs because the BASIC SAVE routine sends CR,LF,CR, LF,CTRL/Z to tape at the end of a file and the VERIFY routine only recognizes CR,CR. To make BASIC files terminate <3> therefore an extra CR must be placed at the end of a program and a method for doing this is as follows:

1. Make the very last line of your BASIC program:
   XXXXX END: (Note: omitting the colon leads to SN ERROR IN XXXXX)

2. ESCAPE to the MONITOR and find where in memory the final colon is using:
   <M> = 75 ab cd XX XX

3. Now look at the memory constants at cdab-4 and you will see:
   <M> = cdab-4 3A ØØ ØØ ØØ

4. Using </> change this to:
   <M> = cdab-4 3A ØD ØØ ØØ

5. Return to BASIC using <6> and SAVE the program in the normal way. It now has CR,CR,LF,CR,LF,CTRL/Z sent to tape and the first two CR's terminate <3>. (On reloading to BASIC this added ØD is stripped off and replaced by ØØ, however it is still on the tape!)

Yours sincerely,
Dr. P.R. Coward
60 Onslow Gardens
Ongar, Essex, England

Dear Editor:

Please inform your readers that I have self adhesive labels available (for use on AIM-65 Keyboard) for use with BASIC TIME SAVER or BASIC SHORT CUT programs. These labels are white with black lettering. (Note that labels should be covered with transparent tape for long term protection.)

USA & CANADA

Send $1.00 per label with a S.A.S.E. max of 4 labels for one stamp.

ELSEWHERE

Send $1.00 per label with a self-addressed envelope plus $1.00 shipping & handling for 1–3 labels; $1.00 shipping for each additional 3 labels.

Ron Riley
Box 4310
Flint, MI 48504

Dear Editor:

I found Ken Fullbrook's advice on using the editor to create basic programs very useful. A variation of this method will allow basic to input data from the editor.

1. Allocate memory space for concurrent usage of basic & editor. Only re-enter basic with 6 key.

2. Create data file in editor, exiting with a "T" & "Q" command.

3. Load in the basic program.

4. The "input" statement that is to use a data element created in the editor must be proceded by the commands:

   POKE 42002,ASC("U")
   POKE 264,208:POKE 265,250

   The first command makes the input come from a user defined subroutine whose location is specified by the second command as residing at hex location FADO.

5. After the input statement(s), follow with this:

   POKE 42002,13

   This returns you to normal input

Keep in mind that if you wish to re-run the program using the editor data file, you must exit basic & reset the editor pointer to the top. Be careful of the number of elements in the data file versus the number of elements you are trying to input. Failure to observe these points usually causes me to have to resort to a re-start.

Michael Chin
Richmond, CA

AIM 65 MONITOR ROM BIT PATTERNS

I want to thank all who responded with programs and/or tables of ROM bit patterns. I'm glad to know there are so many who realize the value of such a thing. The table here was generated with a program written by G. E. April of Ecole Polytechnic (Montreal, Canada).

00 @ E0CD	01 @ E101	02 @ E076	03 @ E0B0	04 @ E079	05 @ E245	06 @ E15C	07 @ E0BA
08 @ E0AB	09 @ E127	0A @ E1AD	0B @ E527	0C @ E0E4	0D @ E203	0E @ E0C8	0F @ E223
10 @ E0A8	11 @ E19A	12 @ E225	13 @ E103	14 @ E096	15 @ E0FB	16 @ E33C	17 @ E134
18 @ E2CE	19 @ E4A8	1A @ E48B	1B @ E205	1C @ E23D	1D @ E242	1E @ E31B	1F @ E313
20 @ E008	21 @ E07C	22 @ E084	23 @ E087	24 @ E093	25 @ E08B	26 @ E08F	27 @ E1E9
28 @ E6C8	29 @ E159	2A @ E009	2B @ E721	2C @ E11D	2D @ E368	2E @ E09C	2F @ E0A2
30 @ E152	31 @ E1DC	32 @ E1D0	33 @ E1DE	34 @ E1DF	35 @ E1E0	36 @ E1E1	37 @ E262
38 @ E16F	39 @ E1E5	3A @ EA55	3B @ E05E	3C @ EF18	3D @ E064	3E @ E18F	3F @ E1D5
40 @ E29F	41 @ E011	42 @ E027	43 @ E04B	44 @ E044	45 @ E01F	46 @ E000	47 @ E1C8
48 @ E04C	49 @ E02A	4A @ E58D	4B @ E063	4C @ E039	4D @ E003	4E @ E022	4F @ E002
50 @ E00E	51 @ EA4C	52 @ E001	53 @ E00F	54 @ E005	55 @ E02E	56 @ E0DE	57 @ E046
58 @ E014	59 @ E017	5A @ E1D9	5B @ E1E2	5C @ ECE2	5D @ E1E3	5E @ E1E4	5F @ E8D7
60 @ E119	61 @ E180	62 @ F313	63 @ F2F9	64 @ F3C6	65 @ E20B	66 @ E31E	67 @ ED90
68 @ E07E	69 @ EA52	6A @ EC2A	6B @ E0AD	6C @ E075	6D @ E0BD	6E @ E316	6F @ E883
70 @ E122	71 @ E460	72 @ E142	73 @ E624	74 @ F0CE	75 @ EE41	76 @ EF6B	77 @ EC3B
78 @ E0C0	79 @ F060	7A @ E188	7B @ E14B	7C @ E13C	7D @ E1B3	7E @ E1B9	7F @ E0B2
80 @ E0D8	81 @	82 @ E1AA	83 @ E8C9	84 @ E3DB	85 @ E265	86 @ E26B	87 @ E476
88 @ E0A5	89 @	8A @ E146	8B @ E51B	8C @ E086	8D @ E07B	8E @ E083	8F @ EC47
90 @ E0AF	91 @ E83A	92 @	93 @ E2EA	94 @ E219	95 @ E3EF	96 @ E18B	97 @ FFA3
98 @ E2CD	99 @ E09B	9A @ E0C3	9B @	9C @ EF15	9D @ E0CC	9E @ E211	9F @ F49B
A0 @ E023	A1 @ E38C	A2 @ E0C1	A3 @ E448	A4 @ E077	A5 @ F54A	A6 @ E3DE	A7 @ E453
A8 @ E0CE	A9 @ E0FE	AA @ E14E	AB @	AC @ E095	AD @ E098	AE @ E1A5	AF @ E22D
B0 @ E21E	B1 @ E7AD	B2 @ F525	B3 @ F576	B4 @ F60A	B5 @ EE73	B6 @ F674	B7 @ F6B9
B8 @ FB35	B9 @ E2DD	BA @ E091	BB @	BC @ E186	BD @ E004	BE @ E2E0	BF @ E020
C0 @ E2B6	C1 @ E069	C2 @	C3 @ E36B	C4 @ E06A	C5 @ E053	C6 @ E026	C7 @ E75C
C8 @ E25C	C9 @ E2AE	CA @ E0CF	CB @ E029	CC @ E067	CD @ E0E0	CE @ E047	CF @ E056
D0 @ E0E3	D1 @ E3E6	D2 @ E054	D3 @ E01B	D4 @ E19F	D5 @ F85E	D6 @	D7 @ E715
D8 @ E082	D9 @ E20D	DA @ FB4B	DB @ E2A4	DC @ E95D	DD @ E196	DE @ F12B	DF @ F6A1
E0 @ E0FA	E1 @ E0B3	E2 @ E0BE	E3 @ E308	E4 @ E1F4	E5 @ E1B0	E6 @ E0A6	E7 @ E0AE
E8 @ E0F9	E9 @ E0BB	EA @ E140	EB @ E258	EC @ E5B7	ED @ E336	EE @ E1F9	EF @ E14C
F0 @ E0AA	F1 @ E25F	F2 @ E1FB	F3 @ E370	F4 @ E151	F5 @ E0FD	F6 @ E1E6	F7 @ E0D1
F8 @ E0B8	F9 @ E123	FA @ E201	FB @ E12D	FC @ E991	FD @ E307	FE @ E13D	FF @ E0C2

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