SYNTAX 2.3

JULY – AUG 1986

TABLE OF CONTENTS

FCAUG

FIRST CANADIAN ADAM USERS' GROUP
P.O. Box 547 Victoria Station
Westmount, P. Q.
H3Z 2Y6

POSTAGE PAID
Mtl. PQ
Permit
7152

L E T T E R S

Dear FCAUG,
One day when I was working with my computer, I noticed that the printer did a backspace. Puzzled I looked to what I had done and found that some ASCII characters can control the printer. For the benefit of other ADAM users, I have included with this letter a small program giving all the possibilities that I found.

```
10 PR #1: PRINT CHR$(13)
20 PRINT "  8=> example of backspace ==="; CHR$(8); "<=="
30 PRINT " 10=> example of"; CHR$(10); "line feed"
40 PRINT " 11=> example of"; CHR$(11); "half line feed"
50 PRINT " 13=> example of"; CHR$(13); "carriage return"
60 PRINT "14-15 example of backward and forward writing        ";
   CHR$(14); "<<<<<<<"; CHR$(15); ">>>>>>>"
70 PRINT " 27=> example of character not found on keyboard ";
   CHR$(27); CHR$(27); CHR$(27)
80 PRINT "127=> example of character not found on keyboard ";
   CHR$(127); CHR$(127); CHR$(127)
90 PR #0
```

Jocelyn Doire
Gatineau, Que.

Dear FCAUG,
Just a couple of paragraphs to tell you that I enjoy your bulletin very much. I typed in the HEXAGON program from the last issue and found it really beautiful - quite something. I played around with it changing a line here and there and adding some randoms now and then. The results are really remarkable. I used carbon paper to print out the programs and the letter. I just put the carbon on the letterhead and type away with no ribbon in the printer. It saves on ribbons. Another thing I like to do is to put around 10 programs under one name. This way I can get about 200 programs on a data pack using 35 file names. I tried to free up the space on my E-Z COPY backup but couldn't do it. Is there any way to do it?

Nicholas Lukach, C.A.
Noranda, Que.

Glad to see that you are enjoying the newsletter, Nick. Thanks for passing on the tip about using carbon paper as a substitute for a ribbon. Those of you who do a lot of program debugging might find this particularly useful for printing out rough drafts of long programs. Your second tip on tying together several small programs is one way of coping with the limit of 35 files per data pack or disk (for another method see the article on multiple block directories elsewhere in this issue). To join several programs together and still have them excecute independently, you must add a small routine at the beginning of the combined programs which allows you to select and jump to the start of the desired program. The individual programs need to be renumbered as if they were a single program. In some cases, it might also be

necessary to add an END statement to each program. Due to the way E-Z COPY is written, it is not possible to recover empty blocks from the tape or disk without affecting the program.

Dear Mr. Moore,
I just received SYNTAX 2.2 and it looks really good. It certainly has a lot of useful information for ADAMites.
I felt the review of our Entertainment Pack was well done, but I do question a couple of things. You mention that Slide Puzzle "is great for very young children who are just starting to count." I think this game is quite difficult (especially if you haven't played it a lot) and can't quite picture too many kids solving it. Another thing that bothered me was that although the whole review sounded positive, the last sentence reads, "anyone looking for more games won't be too disappointed." To me this is a negative conclusion saying that I, as the user, will be a little disappointed (but not much). Overall, I thought it was a good review though. Maybe I'm just nit-picking on these two points.

Jack Reedy
REEDY SOFTWARE

I regret for neglecting to quantify the phrase, "very young children." To me, this includes children in the 5-8 age group. The statement is not meant to question the difficulty level of the game but to imply that these are the users that will derive maximum satisfaction and challenge from solving the puzzle. The educational value of the program and the superb graphics are certainly distinguishing features of this fine game.
It was not my intention to end the review on a negative note but to conclude that this set of game programs as adaptations of familiar games will have a somewhat limited appeal. This is not to be misinterpreted as a knock on the games as their superiority was clearly highlighted throughout the review.

Dear FCAUG,
Thank you for all the interesting and informative articles in SYNTAX. When we bought ADAM we had little or no experience with computers. SYNTAX has provided us with much valuable information. A couple of books that have helped us get started with our computing are: 1. How to Use The Coleco ADAM written by Jerry Willis and published in 1984 by Dilithium Press 8285 S.W. Nimbus, Suite 151, Beaverton, Oregon 97005. This is an easy to understand how-to book. It covers everything from a description of ADAM as it is straight from its carton through set-up, word-processing, Basic programming (including graphics), plus one chapter on selecting hardware and software and another on sources of information. It compliments nicely the manuals that came with ADAM.
2. The Family Home Computing Library (five volume set) printed in Italy in 1984 by Orbis Publishing Ltd., London and distributed by Heron Books, 94 Crockford Blvd., Scarborough, Ont. M1R 3C5. These books are set up in lesson form and are written for most if not all PC's. It reviews many of the more popular home computers in a hardware section. It also reviews software and contains sections on Insights, Feedback (question/answer format), Basic

programming, Passwords to computing, Sound and Sight plus a few more. It covers a lot of material in an easy to understand way and should be a tremendous help to beginners.

J.E. Tessari
Milk River, Alta.

Dear FCAUG,
I have been greatly impressed with SYNTAX this past year. I have found many of the articles interesting and valuable as I struggle along with discovering the many capabilities of my ADAM. I appreciate the time and effort required to produce such a quality publication.
When I wish to separate lines of text using SmartWriter, (such as between two paragraghs), I usually type two returns - one to end the line and one to leave a blank line. This actually results in 1-1/2 spacing when printed out by the printer. This is usually acceptable; however, there are times when I only want one space between lines of text. To overcome this, when necessary, I have forced a single blank line by typing a period and then the return. I then white out all the periods after printing. This is very inconvenient and messy. Do you know of any way of getting a single blank line between lines of text?

Chris J. Allan
Sleeman. Ont.

The 1-1/2 spacing bug is a familiar one to those users who do a lot of word-processing. Entering a single character before the carriage return will give you the correct single spacing for a blank line. This is the easy method most of us have adopted to cope with the problem. There is a way to correct for it. First make sure that the line preceeding the place where you want a blank line inserted starts at the leftmost position of your selected margin. Now using the SUBSCRIPT Smart Key enter a space with the spacebar. Hit the DONE Key and enter a carriage return. This procedure will eliminate the unwanted extra blank half line.

Dear Syntax,
I notice that most of the articles are slanted toward the programming buffs and so it should be as there are more of them. I bought my ADAM primarily for the word processing feature and to date it does just about everything I require. There are a few exceptions which I hope you can clear up. When using fan fold paper in the printer, I find that the print seems to creep down the page and after three pages we have the type ending very close to the fold. It gets progressively worse after each page until it runs onto the next page. It's as if the counter is out. Would a tractor feed correct this problem or is it a quirk of the Adam?

Bill Mayrs
North Vancouver, B.C.

The problem you are describing seems to be related to the spacing problem referred to in the previous letter. The extra half spaces

with each carriage return entered to force a blank line in your document accumulate from page to page gradually shifting your top and bottom margins. The effects of these spaces when printing out multiple page documents plus what appears to be an inconsistency with the friction feed mechanism of the printer become more apparent. A tractor feed attachment will not solve the problem as it is the printer mechanism which is responding to the software instructions which will drive it. It will only help to hold the fan fold paper in place and avoid it from slipping and jamming. To determine if the software is indeed the culprit, try printing out a 3 or 4 page document containing no blank lines (i.e. no paragraphs just one continuous text). If your margins are not altered then you know you have found the source of the problem.

Dear Sirs,
There is a printer ribbon available for the ADAM printer for about $10.00 from KO-REC-TYPE, Barough Eaton (Canada) Ltd., 6291 Ordan Drive, Mississauga, Ont., L5T 1G9. It is just like the ribbon supplied by Coleco.

Sami Cokar
Calgary, Alta.

Dear FCAUG,
In an earlier issue of SYNTAX a west coast member mentioned that the Pacific Ribbon & Carbon Co. of Burnaby refills multistrike ribbon cartridges. I've found that the company will do this as a mail service and recently I mailed two cartridges in a padded envelope. They were refilled and returned by parcel post in about two weeks. The cost was $7.00 plus $1.82 for delivery - a total of $8.82. Do not enclose money; the company will send an invoice by first class mail. The address is 3839 Still Creek Ave., Burnaby, B.C. V5C 4E2. Telephone (604) 430-1191.

John H. Michell
Toronto, Ont.

Editor's Note ..... These two alternative sources are just some examples of the many suppliers ready to serve ADAM users.

FCAUG is now making available for $10.00 on disk or tape the following Coleco titles released to the public domain: Troll's Tale, Super Subroc, Hard Hat Mack/Pinball Construction, Jeopardy and SmartBASIC V2.0. Those who ordered SmartBASIC V2.0 at the old price can request any 2 of the above. Next month we begin selling 64K memory expanders, printer interfaces and 80-column cards. Look for details and prices in the next issue or write to us.

SYNTAX is published bimonthly by First Canadian Adam Users' Group (FCAUG), P.O. Box 547, Victoria Station, Westmount, Que., H3Z 2Y6 Subscription Rates: 6 issues for $22.00. Second Class Mail Registration No. 7152. POSTMASTER: Send address changes and notice of undelivered copies to above address. Return Postage Guaranteed.

BASIC PARTS - DATA and REM STATEMENT BUG

A very annoying bug we all have experienced, and which is now history thanks to SmartBASIC V2.0, is that of the growing DATA and REM statements. Every time you edit a DATA or REM line in your program, SmartBASIC V1.0 will add another space between the command and the input. If you make many changes and ignore these extra spaces eventually some of your input will be lost at the end of the line. With DATA statements, if some of your DATA is pushed off at the end of the line your program will not execute properly. This bug also surfaces every time you LOAD a program already SAVEd and re-SAVE it.

One solution to this problem is to POKE the following memory locations with these new values: 15824,216 : 15830,8: 15831,55: 15832,19. Remember that these changes are not permanent and will be lost when you shut off your unit. You can put them in a HELLO program so that they will be made every time you load SmartBASIC.

As is often the case, there is more than one way to correct the problem. The best solution is to modify the Basic program itself. This requires some knowledge on editing specific blocks on a data pack or disk. The patch below was written by Bob Hamilton of the Vancouver Adam Club (V.A.C.). Note that it does not work with "&" when used instead of REM whereas the poke method works with both. You will need a Tape/Disk Editor program to modify block #2 of your SmartBASIC program. Make the changes on a backup copy only. For this purpose you can use any one of these utilities: Uncle Ernie's, E-Z EDIT, JKL or the one by B. Hinkle from the "Hacker's Guide". All the information given below is in hexadecimal.

```
BLOCK 2......PAGE 1......BYTE 4F......WAS B9......CHANGE TO 65
BLOCK 2......PAGE 1......BYTE 50......WAS 03......CHANGE TO 04
BLOCK 2......PAGE 1......BYTE 63......WAS BC......CHANGE TO 68
BLOCK 2......PAGE 1......BYTE 64 .....WAS 03......CHANGE TO 04
BLOCK 2......PAGE 4......BYTE 57......WAS 30......CHANGE TO 2A
BLOCK 2......PAGE 4......BYTE 65......WAS 7A......CHANGE TO 01
BLOCK 2......PAGE 4......BYTE 66......WAS 65......CHANGE TO 6E
BLOCK 2......PAGE 4......BYTE 67......WAS 72......CHANGE TO 04
BLOCK 2......PAGE 4......BYTE 68......WAS 20......CHANGE TO 01
BLOCK 2......PAGE 4......BYTE 69......WAS 4D......CHANGE TO 6B
BLOCK 2......PAGE 4......BYTE 6A......WAS 69......CHANGE TO 04
BLOCK 2......PAGE 4......BYTE 6B......WAS 63......CHANGE TO 06
BLOCK 2......PAGE 4......BYTE 6C......WAS 72......CHANGE TO 3A
BLOCK 2......PAGE 4......BYTE 6D......WAS 6F......CHANGE TO 21
BLOCK 2......PAGE 4......BYTE 6E......WAS 53......CHANGE TO 06
BLOCK 2......PAGE 4......BYTE 6F......WAS 79......CHANGE TO 00
BLOCK 2......PAGE 4......BYTE 70......WAS 73......CHANGE TO 1A
BLOCK 2......PAGE 4......BYTE 71......WAS 74......CHANGE TO FE
BLOCK 2......PAGE 4......BYTE 72......WAS 65......CHANGE TO 20
BLOCK 2......PAGE 4......BYTE 73......WAS 6D......CHANGE TO 20
BLOCK 2......PAGE 4......BYTE 74......WAS 73......CHANGE TO 01
BLOCK 2......PAGE 4......BYTE 75......WAS 20......CHANGE TO 13
BLOCK 2......PAGE 4......BYTE 76......WAS 49......CHANGE TO C3
BLOCK 2......PAGE 4......BYTE 77......WAS 6E......CHANGE TO CB
BLOCK 2......PAGE 4......BYTE 78......WAS 63......CHANGE TO 3D
```

Multiple Block Directories

In SYNTAX 2.1 we briefly discussed how Adam stores information on data packs or disks. We explained why we are limited to a maximum of 35 file entries. Evidently, this does not give us much to work with when you consider that deleted files are also included in this count. What this means is that once you have stored 35 files on a data pack or disk, you will not be able to add anymore even if you delete some. The only way to reuse it is to INITialize it. Below we present one method to avoid this problem. This procedure is based on an article by Dan Grelinger which appeared in the July newsletter of the Kansas Adam Users' Group.

Only a single block is allocated to the directory on every data pack or formatted disk. As we saw in SYNTAX 2.1, all it takes is 35 file entries even if each is only one block long to fill this block. Once it is full, it will not accept anymore files and you get the "No More Room" error message. The source of our problem is precisely this one block limit. Fortunately for us, Adam's operating system can accept multiple block directories. You have to make the changes yourself however as the system has no provision for creating a multiple block directory.

Using a Tape/Disk block editor program you can modify the size of the directory block of a blank data pack or disk. Changing it to 2 blocks increases the number of possible file entries to 74. To calculate the larger size associated with this single block increase refer to the formula given in SYNTAX 2.1. Remember to deduct the blocks occupied by the 4 "resident" programs. Working this out for a 2 block directory, we get the following equation: (1024*2/26 = 78.77). This means that our directory can hold 78 complete files. Now we deduct 4 (the 4 resident files) from this figure to arrive at our new limit of 74 possible file entries.

To make these modifications you must change the following byte values in block #1. All information given is in hexadecimal.

```
BLOCK 1.....BYTE 0C.....WAS 81.....CHANGE TO 02
BLOCK 1.....BYTE 11.....WAS FF.....CHANGE TO 00  (optional)
BLOCK 1.....BYTE 12.....WAS 00.....CHANGE TO 01  (optional)
BLOCK 1.....BYTE 45.....WAS 01.....CHANGE TO 02
BLOCK 1.....BYTE 47.....WAS 01.....CHANGE TO 02
BLOCK 1.....BYTE 5B.....WAS 02.....CHANGE TO 03
BLOCK 1.....BYTE 5F.....WAS FD.....CHANGE TO FC
BLOCK 1.....BYTE 5F.....WAS FD.....REMAINS   FD  (optional)
BLOCK 1.....BYTE 5F.....WAS 9E.....CHANGE TO 9D  (for disks only)
```

The above changes are for a data pack. They also apply to disks with the only difference being that byte 5F is changed from 9E to 9D. The optional changes allow the user to recover an extra block on blank data packs. Do not include these changes when creating a multiple block directory on disk. The procedure outlined above will work only with blank data packs or disks. Tapes containing programs must be INITialized first while used disks should be reformatted.

Helpful Hints, Changes and Corrections
for 32 BASIC Programs for the Coleco Adam

Many early Adam owners received this combination book and data pack as a bonus from a Coleco promotion back in 1984. Several changes and improvements were made by the authors since then. We have decided to reprint them here for the benefit of those who own this software. The book by the same title published by Dilithium Press and written by Tom Rugg and Phil Feldman is still available. It does not include these modifications to the program listings as it was printed prior to their release.

For the benefit of those who have not seen or been able to find these programs, a brief description of the contents follows. This Software Library is a collection of 32 BASIC programs made up of six categories of programs: Applications, Educational, Graphics, Games, Mathematics and Miscellaneous. The book explains the purpose, use, main routines, main variables, and suggests some changes and projects for each program. An illustrated sample run and the complete listing is also included. The programs are excellent examples of SmartBASIC's possibilities and versatility. You can use these examples as a guide for improving your own programming skills.

If you have or plan to get the book you will find that entering the programs yourself can be very time-consuming as most of the programs are quite long. We can provide you with all the programs ready-to-run on a data pack or disk. The cost for this service is $10 and includes media plus postage and handling.

General Operating Notes

If the program cues you to "PRESS ANY KEY" and the key you pressed doesn't work, try another.
If your LOCK key is on, ADAM may not read your commands.
There are several ways to escape a program. If one doesn't work, try another. 1. You can press the CONTROL key and the letter C simultaneously, then hit RETURN. 2. You can press the ESCAPE/WP key. Either method takes you back to SmartBasic.

TACHIST and FLASHCARDS
Do not type any extra spaces when you are typing in answers.

QUESTEXAM
To start the program, press the spacebar or the return key. Once you have completed the program, do not add entries to total a greater number than initially specified. Restart the program from the beginning to add a larger number of entries.

DECIDE
The maximum number of entries is 9.

SIMEQN
The program will not let you divide by zero.

BIRTHDAY
To stop the program, so you read the screen, hold down the control key and press the S key. To resume the program, press the control and S keys again simultaneously.

OBSTACLE
To start the program, press either the spacebar or the return key.

Program Changes

The following programs have been changed and improved since the book was printed.

DECIDE
```
 510 PRINT " I need a list of each";
 520 PRINT t$; "being considered.": PRINT
1220 IF t <> 3 THEN PRINT "Each"; t$; "must"
1350 IF t <> 3 THEN PRINT "Every other"; t$;
1450 PRINT "What value would you assign to";
```

VOCAB
```
1120 IF h >= 5 THEN 1140
```

ARITHMETIC
```
 140 k = RND(2): k = PEEK (64885): IF k <>67 AND k <>99 THEN 140
```

METRIC
```
 140 q = RND(2): q = PEEK (64885): IF PEEK (64885) <>67 AND PEEK
     (64885) <>99 THEN 140
```

WALLOONS
```
2320 x1 = x3: x = x1 -11: rv = 48: GOSUB 260: rv = 32: x = x3 +1:
     y = y2 +32: GOSUB 260
2330 rv = 16: x = x3 +11: y = y2 +26: GOSUB 260: rv = 0: x = x3:
     y = y2 +20: GOSUB 260: NEXT
```

ROADRACE
```
2090 IF PEEK (64885) <>67 AND PEEK (64885) <>99 THEN 2090
```

GROAN
```
 180 q = RND(2): q = PEEK (64885): IF q <>67 AND q <>99 THEN 180
```

JOT
```
 220 q = RND(2): q = PEEK (64885): IF q <>67 AND q <>99 THEN 220
```

SIMEQN
```
 305 IF n <> INT (n) THEN PRINT: PRINT "**ERROR!** IT MUST BE A
     WHOLE NUMBER": PRINT: GOTO 300
 310 PRINT: IF n > 0 and n <= m THEN 340
```

PI
```
 140 IF PEEK (64885) <>67 AND PEEK (64885) <>99 THEN 140
```

CP/M Corner - Review: MEX - A Modem Executive Program

On first contact with MEX, one can feel a little overwhelmed. There are so many commands and options that you cannot imagine any practical use for many of them. But MEX is more than just a complex program. It has a lot of substance, as we shall soon see.

Once you start using the program you realize that it is extremely sophisticated. One of the things that makes it so interesting is its high degree of programmability. From the command mode, you can send any number of lines of instructions from a special batch file with one or two words.

You start by writing this special instruction file with any text editor (WordStar in non-document mode) before running MEX. This file should contain a series of commands that can be executed from the command mode. The filename must end with the extension ".MEX". A special command, SENDOUT is provided that allows you to send out a string, enclosed in quotes, to the other (remote) computer. SENDOUT is like being in a temporary terminal mode without leaving the command mode. What is the advantage to all this? You can use .MEX batch files to set up various parameters, automate log-on procedures, or even download files from a BBS completely unattended. And like the resident CP/M file SUBMIT you can create general purpose .MEX files by using variables in your text. To illustrate the point, let's say you want to download a file named TEXT.DOC with your .MEX file called DOWNLOAD.MEX. Just write a {1} where the name of the file to be downloaded would normally go. To execute the sequence type READ DOWNLOAD TEXT.DOC. You can add as many variables as you like. The variable names will be filled in the order that they appear after the .MEX file. I hope that you can see the potential for a feature such as this.

MDM7 users will be glad to know that MEX shares many features with that program. In fact, MEX is based on MDM7 and is, as its author puts it, "a superset of such programs as MODEM2, MODEM7, and MDM". So if you know how to use MDM7, MEX should not be too hard to grasp. There are little differences like using ^S instead of ^Y to start the terminal capture, but these are easily surmounted. There are however a few differences that can lead to confusion for the MDM7 user. What follows are two of the major stumbling blocks that I have encountered.

Trying to exit the terminal mode for the first time was, well... trying! Thinking all I had to do was press ^E to get to the command mode was a wrong assumption. MEX wants first an escape character which puts you in an intermediate mode (limbo) and then "e". If you screw up here you remain in limbo. The escape character for most people is "0A" or "^J". You can change this character to anything you like and that is exactly what I did. Mine is now "1B". What this means is that the ESCAPE/WP key now gets me where ^J used to, and I find this a little easier to remember and implement.

The secondary command "A" has been changed. Instead of being

(a)nswer it is now (a)ppend. This is, for me, a drawback. With MDM7 you can call another user and he/she can then (a)nswer you with his/her modem. Or you can switch back and forth between the phone and modem, reconnecting to the modem with the secondary commands O and A. MEX has traded this feature for an (a)ppend text to an already existing file command and frankly I don't see the advantage.

MEX has a plethora of commands that are useful rather than redundant, as I mentioned they may appear at first. Many of these redundancies are related to the .MEX batch files. For example, the command DIAL does the same thing as CALL so why have it? Because DIAL does not put you in terminal mode when it connects. You would use this in a .MEX file because by putting you in the terminal mode, CALL would stop execution of the file. Remember that .MEX files cannot operate from the terminal mode.

There are many functions for you to discover on your own but here are some of the more important ones. MEX has an extensive on-line HELP file. If you get stuck anywhere, typing "HELP" plus the file you want help on should get you out of it. The phone and definable keystring libraries can be edited and accessed from within, unlike M7LIB and M7FNK with MDM7. Besides the normal XMODEM file transfer protocal, MEX has a CIS protocal for Compuserve users. Since I have not used this service, I cannot comment on the use of this feature. The STAT command allows you to get status on various variables within your control. (eg. STAT CIS tells you if the Compuserve protocal is operative or not and following these two words with ON or OFF will change its status.)

The documentation is quite thorough but I have noticed a few mistakes. In particular, the 5 ways that MEX can hang up come to mind. You can either hang up the line but stay in MEX (DSC), keep the line and exit MEX (CPM, EXIT, SYSTEM), or hang up the line and exit the program (BYE). The documentation is wrong for most of these commands (so go by this text) and even if it were right what good is DSC if you can't get back on-line. As I said, I often switch from phone to modem and back again. Obviously MEX was written solely as a program to communicate with BBS's and not as I use MDM7.

The version of MEX that I have is 1.14 and was written by Ron Fowler. It is the latest one that I am aware of. I mention this because Adam users must have this version plus a version of the Adam overlay. There are two overlays that I know of; V1.0 and V1.14. The latter was written by Cruz Controls and is the most recent. This overlay uses the programmable Smart Keys to operate the most often used commands. The dialing option is super fast but doesn't seem to work on all systems (like mine!). I am sure that this is a problem with my line, however, as MEX dials properly on every other Adam I've tried it on. A further note on the MEX version is that Mr. Fowler is planning to improve his program, but may be charging money for it. The obvious question is, what improvements can he make to a program that can do just about everything already? We will have to wait and see.

A D A M G R A P H I C S

Yes, it's that time again, time for another game. In this issue, I will explain a game that uses sprites and a shape table in the Hi-Res graphics mode.

The game is called Outfielder and is loosely based on baseball. A ball is randomly hit into the outfield and you, as the outfielder, must try to catch it. You are randomly placed in the outfield which is surrounded by four walls. If the ball gets past you, it will continue to bounce off the walls until you grab it. If the outfielder catches a pop fly, the batter is out. The base runners will keep running even if the pop fly is caught. So if there are men already on base you must throw the ball to one of the 4 bases and try to tag the runner out. If men are half way between bases, they will attempt to get to the next one. This is the only chance you have to throw one of the runners out.

The game starts by asking you how many outs you want to get. Try entering 27, the number required in a 9 inning baseball game. You enter the number of outs from the keyboard. All other input will be from paddle number one. To start the game press one of the fire buttons. The ball will move out from home plate to somewhere in the outfield. You can move the outfielder (a glove) by using the joystick. To field the ball you only need to come in contact with it. You throw the ball to a base by pressing one of the keypad numbers. Pressing number 1 through 4 sends the ball to the corresponding base. Home plate is designated as base #4. The closer you are to a base, the sooner the ball will reach that base. A scoreboard displays the number of runs scored as well as the number of outs.

In designing the game, I wanted the ball and glove (both sprites) to move as fast as possible. Unfortunately, SmartBASIC proved to be too slow. To make the ball and glove move faster I made them move in four pixel increments. This accounts for their somewhat jumpy appearance. To make the game a little harder, I decided to randomly place the outfielder in one of three positions corresponding to the three outfielder positions in baseball. Before the ball is hit, the player has no idea where he will be placed thus giving some added difficulty to the game. I chose to make the base runners Hi-Res shape drawings rather than sprites in order that they would not interfere with the sprite collision detection between the ball and glove. This decision causes slower execution of the program. You will notice a slight pause in the movement of the sprites when a base runner is moving, but this adds to the difficulty of the game. Having only two sprites simplifies the coding of the game thanks to a register in the Video Display Processor (VDP) which notes when any two sprites come in contact. Therefore, all one needs to do is to check a particular bit in this register to see if two sprites have collided. In an earlier article, I wrote a short machine language routine to do this but in this game I have found an easier way. The operating system has a routine that returns the VDP register we want. The nice thing about the routine is that it leaves a copy of the VDP status

register in memory location 64867 (FD63 hex). Now all we have to do is call this routine from BASIC (CALL 64803) and peek location 64867 for the status register. A value of 162 at this location means a sprite collision.

You may notice that many of the lines contain a lot of code. This was done purposely to speed up the execution of SmartBASIC since having many statements on one line number means less work for SmartBASIC. When typing in a long line you should skip typing in any spaces that aren't necessary, this includes the space between the line number and the first statement. Rather than typing in "PRINT" use the question mark "?" instead. To append a statement to an already long line, list the line and move the cursor to the beginning and go over, using the left cursor, all the vital text. If you come up to a space that isn't necessary you can skip over it by holding down the control key and the right cursor key at the same time. Doing this will allow you to pack in more text per BASIC line. You should type in the program using the BASIC editor rather than SmartWRITER since the latter likes to truncate long lines. (See SYNTAX 1.1 for more SmartBASIC editing tips.)

EXPLANATION OF PROGRAM LINES

LINE #	COMMENT
110	leave some room for machine code routines and shape table, define matrices to hold base runner coordinates and base coordinates.
120	allow the poke function to operate in all of memory and select a green background for the Hi-Res graphics mode.
140-150	load a machine language routine at 28000 to write N bytes from the buffer at 28013 to an address in video memory. The number of bytes to write is poked in at location 28007, the video address is poked in at locations 28001 and 28002. The data to be written is poked in the buffer.
160-200	poke in video address of sprite pattern table which is stored at locations 64870 and 64871, write data to video memory.
210-230	poke video address of sprite attribute table which is stored at locations 64868 and 64869.
240-260	put base runner shape table at location 28050.
270-320	initialize various matrices.
330-380	draw baseball diamond and outfield walls, get number of outs from the user.
390-420	place ball and glove in their initial positions, randomly select the column location of the glove, write both sprites at same time.
440-460	randomly choose how often to move ball vertically as well as how far to move it.
470	move ball vertically.
480	allow ball to move through top wall once, check for first bounce off wall.
490-510	count off delay for ball's X movement, move ball horizontally if delay reaches zero.

```
100 &  **OUTFIELDER** - Andrew Wiles (1986)
110 LOMEM :28100: DIM m(1, 12), bas(1, 4)
120 POKE 16149, 255: POKE 16150, 255: POKE 25431, 12: POKE 25471, 0: HGR
130 &   load up memory and screen tables
140 FOR i = 0 TO 12: READ d: POKE 28000+i, d: NEXT
150 DATA 17,0,0,33,109,109,1,0,0,205,26,253,201
160 POKE 28001, PEEK(64870): POKE 28002, PEEK(64871)
170 POKE 28007, 16: FOR i = 0 TO 15: READ d: POKE 28013+i, d: NEXT
180 CALL 28000
190 &   ball and glove pattern bytes
200 DATA 96,240,240,96,0,0,0,0,195,231,231,255,255,255,126,60
210 POKE 28001, PEEK(64868): POKE 28002, PEEK(64869)
220 POKE 28007, 9: FOR i = 0 TO 8: READ d: POKE 28013+i, d: NEXT: CALL 28000
230 DATA 1,126,0,15,133,124,1,9,208: & attribute table data for 2 sprites
240 POKE 16766, 146: POKE 16767, 109
250 FOR i = 0 TO 14: READ d: POKE 28050+i, d: NEXT
260 DATA 1,0,4,0,53,53,63,60,54,45,53,59,46,5,0: & base runner shape table
270 FOR i = 0 TO 12: READ d, dl: m(0, i) = d: m(1, i) = dl: NEXT
280 FOR i = 0 TO 8: READ d, dl: r(0, i) = d: r(1, i) = dl: NEXT
290 DATA 0,0,0,-1,1,0,1,-1,0,1,0,0,1,1,0,0,-1,0,-1,-1,0,0,0,0,-1,1
300 DATA 180,0,95,9,62,31,97,52,128,70,158,51,193,32,161,12,127,6
310 FOR i = 1 TO 4: READ d, dl: bas(0, i) = d: bas(1, i) = dl: NEXT
320 DATA 64,40,128,80,192,40,128,0
330 HCOLOR  = 3: HPLOT 128, 0 TO 0, 80: HPLOT 128, 0 TO 255, 80
340 HPLOT 0, 80 TO 255, 80: HPLOT 0, 158 TO 255, 158: HPLOT 0, 80 TO 0, 158
350 HPLOT 255, 80 TO 255, 158: HPLOT 64, 40 TO 128, 80
360 HPLOT 192, 40 TO 128, 80: INPUT "number of outs? "; mout
370 SCALE  = 1: ROT  = 0: HCOLOR  = 4
380 HOME: PRINT "    OUTS 0"; : HTAB 20: PRINT "RUNS 0": GOTO 790
390 rc = 20: p(0) = 1: DRAW 1 AT r(0, 0), r(1, 0)
400 xg = 188-INT(3*RND(1))*64: yg = 133: y = 1
410 POKE 28007, 6: POKE 28013, y: POKE 28014, x: POKE 28017, yg
420 POKE 28018, xg: CALL 28000: CALL 64803
430 b = 0: y = 1: x = 126: mes = 0
440 xo = INT(5*RND(1))+1
450 xd = INT(((xo > 3)+(xo = 3)*3+(xo = 2)*4+(xo = 1)*10)*RND(1))+1
460 xo = xo*(1-2*INT(2*RND(1))): xc = xd: yo = 4: f = 0
470 o = yo: yo = yo*(1-2*((y = 81 AND f)+(y >= 153))): y = y+yo
480 f = f+(f = 0 AND y = 85): b = b+(o <> yo AND b = 0)
490 xc = xc-1: IF xc = 0 THEN  xc = xd
500 o = xo: xo = xo*(1-2*((x < 5)+(x > 249)))
510 x = x+xo: b = b+(o <> xo AND b = 0)
520 POKE 28013, y: POKE 28014, x
530 p = PDL(5): xg = 4*m(0, p)+xg
540 yg = m(1, p)*4+yg: xg = xg+((xg < 0)-(xg > 250))*4
550 yg = yg+((yg < 80)-(yg > 152))*4
560 CALL 64803: IF PEEK(64867) = 162 THEN 580
570 POKE 28017, yg: POKE 28018, xg: CALL 28000: GOSUB 810: GOTO 470
580 HTAB 1: VTAB 23: PRINT "THROW TO BASE 1, 2, 3, or 4"
590 p = PDL(13): FOR i = 1 TO 60: NEXT
600 IF p(2) = 0 AND p(4) = 0 AND p(6) = 0 THEN  GOSUB 810
610 IF b = 1 THEN 660
620 i = -1
630 i = i+1: IF p(i) = 0 THEN 630
640 p(i) = 0: XDRAW 1 AT r(0, i), r(1, i): out = out+1: b = 1
650 HTAB 10: VTAB 21: PRINT out; : mes = 1
660 IF p < 1 OR p > 4 THEN 590
670 delay = INT((ABS(bas(0, p)-xg)+ABS(bas(1, p)-yg))/15)+1
680 FOR i = 1 TO 60: NEXT
690 IF p(2) = 0 AND p(4) = 0 AND p(6) = 0 THEN  GOSUB 810
700 delay = delay-1: IF delay <> 0 THEN 680
710 POKE 28013, bas(1, p): POKE 28014, bas(0, p)-2: CALL 28000
720 IF p(p*2-1) THEN  p(p*2-1) = 0: XDRAW 1 AT r(0, p*2-1), r(1, p*2-1): out =
out+1: HTAB 10: VTAB 21: PRINT out; : mes = mes+2
730 rc = 1: IF p(2) = 0 AND p(4) = 0 AND p(6) = 0 THEN  GOSUB 810
740 HTAB 1: VTAB 23: PRINT CHR$(24)
750 IF mes > 1 THEN  HTAB 1: VTAB 23: PRINT "MAN OUT AT BASE "; p;
760 IF mes = 1 OR mes = 3 THEN  HTAB 20: VTAB 23: PRINT "POP UP";
770 IF mes <> 0 THEN  FOR i = 1 TO 3000: NEXT
780 IF out >= mout THEN  HTAB 1: VTAB 22: PRINT SPC(10); "GAME OVER": END
790 HTAB 1: VTAB 23: PRINT "PRESS FIRE BUTTON FOR HIT ": FOR seed = -99999 TO
-1: IF PDL(7) OR PDL(9) THEN  n = RND(seed): GOTO 390
800 NEXT: GOTO 390
810 rc = rc-1: IF rc <> 0 THEN 850
820 rc = 20: FOR i = 7 TO 0 STEP -1
830 IF p(i) THEN  XDRAW 1 AT r(0, i), r(1, i): p(i) = 0: p(i+1) = 1: DRAW 1 AT
r(0, i+1), r(1, i+1)
840 NEXT: IF p(8) AND out < mout THEN  p(8) = 0: XDRAW 1 AT r(0, 8), r(1, 8):
rn = rn+1: HTAB 25: VTAB 21: PRINT rn; CHR$(7)
850 RETURN
```

520	poke new x and y coordinates of ball in buffer.
530-550	move glove with paddle.
560	check for sprite collision.
570	write both sprites to video memory.
580-660	the ball has been fielded, wait for player to throw the ball to one of the bases, in the mean time continue moving the base runners.
670	player has thrown the ball to a base, calculate the delay for the ball to reach the base, increasing the divisor (currently 15) will speed up the throw.
680-700	continue moving the base runners until the ball reaches the base.
710	the ball reached the base, place ball at the base.
720-730	check for a man out at base.
740-760	display messages, PRINT CHR$(24) erases from the cursor to the end of the screen.
770-780	check the number of outs, quit if enough.
790-800	wait for user to press key to start next ball moving in the mean time count down a seed to be used by the random function.
810-850	move base runners, update runs scored.

- - - - - - -

Hardware Review: PIA2 Centronics Printer Interface
Manufacturer: Orphanware - Akron, OH

ADAM is finally coming of age. Last issue, you may have noticed that our mailing labels were printed with a dot-matrix printer. What you may not have realized was that these labels were printed with the ADAM computer.

Previously the ADAM system - good as it is - was a completely closed system. This meant that only ADAM modems, disk drives, printers, etc., could be used with it. Although these items are good, they are often hard to find. Wouldn't it be nice to have an alternative to this very limited source. Well finally, this situation is about to change for all ADAM users and especially those in Canada.

Orphanware is the first major third party manufacturer of hardware for the ADAM computer that will have its products widely distributed in Canada by FCAUG. We will also carry their 64K (128K & 256K models later on) memory expanders as well as the soon-to-be-released 80-column card. Complete details and prices will be printed in the next issue. Inquiries on this subject are welcomed. Now through First Canadian AUG, you will be able to buy the Orphanware Centronics (or parallel) printer interface. With this device, you can use almost any other printer with your ADAM.

Here's a look at what you get. Included with the interface card is a tape or disk which contains the operating instructions and the programs that you need to use it. The interface resembles the 64K memory expander and is about the same size and shape. It is inserted into the second expansion slot inside the memory console of your ADAM. In addition to this, you must buy a printer cable

like the Radio Shack Model 100, part #26-1409, which retails for about $19.95. This ribbon connector slides very neatly out from under the back of your memory console cover.

Since this is new territory for most ADAM users, let me backtrack a bit and explain about printer interfaces and such. Of course, the whole point of buying this package is to be able to attach a different and perhaps superior printer to your present ADAM system. This means that many printers made by other manufacturers can now be made to work on the ADAM if you provide the right connectors or interface. This is precisely what Orphanware has done. But before you go out and buy your printer be sure that it comes with a parallel interface (Centronics) connector. Most do, and some even come with a serial or RS-232 interface. These two types of interfaces are the standard that pretty well all computers use today. If a printer you have access to (such as the one at the office) has a parallel interface connection, then the chances are that you can now use it with your ADAM and it will work properly. You can even upgrade your ADAM to a very fast dot matrix printer if you wish.

And now we get to the fun part - using the interface. You simply boot the tape or disk and in a matter of 2 seconds flat, the software that redirects the print through the parallel printer card and into the new printer is loaded. If you want to use the electronic typewriter you hit the <ESCAPE> key. A second touch of that key, will put you into the word-processing mode. If you want to run a program, you insert the tape or disk and then type in the number that corresponds to the desired drive. It's as simple as that. All this information is displayed on the main screen making it easy to use. But the best feature is that it's all done very fast.

If you want to use your new printer with CP/M you must first create a special CP/M file. Orphanware provides you with step by step instructions. You then run this file and all your writing is redirected to the new printer. For the technically oriented, this file changes the I/O byte from LPT1: to UL1:.

The only complaint I have with the interface is that it doesn't allow the printer to handle the underline function very well. My printer goes all out of wack, so I just don't print any underlined text with it. Later revisions of the software will probably address this problem. In the mean time, this is a small price to pay for such a valuable device and our first real window to the outside computer world.

USERS' CORNER - Wanted fellow users to communicate with.

R.B. Malcolm
32 Wedgewood Dr.
Rothesay, N.B.
E0G 2W0

Jean Laflamme
12560 Beauchatel
Montreal, Que.
H1E 4A5

Murielle Maynard
P.O. Box 1069
Stonewall, Man.
R0C 2Z0

Machine Language Primer

This article presents a machine code to mnemonic table for the entire Z80 CPU instruction set and describes the various addressing modes. When you learn what these mnemonics do you can use this table to convert assembly language programs to machine code.

An assembler mnemonic can be broken into two parts, the opcode and the operands. The opcode is the operation that the instruction is doing while the operands are what it's doing it on. For example, to add 3 to register A, we use the mnemonic ADD A,3. ADD is the opcode while A and 3 are the operands. The word "opcode" is often used to describe the instruction machine code. Therefore an opcode to mnemonic table is the same as a machine code to mnemonic table.

Some instructions have no operands like RET which is used to return from a subroutine. Others appear to have only one like SUB B, - subtract the contents of register B from the contents of register A. Register A is implied in these types of instructions. Most instructions have two operands.

Some instructions like ADD B,C use only registers but many use memory locations. The most often used type of memory addressing is where the second operand immediately follows the opcode in memory. ADD A,3 is an example of this. The byte value 3 follows the ADD instruction. The immediate value may also be a 16 bit (2 byte) value if the first operand can handle it (i.e. a 16 bit register). This mode is known as immediate addressing. If an operand has parenthesis around it then it is assumed that a 16 bit address is stored there. The value at the address in the parenthesised operand is used in the instruction. This is known as indirect addressing. For example, if register BC has the address 60000 in it and the value at location 6000 is 12 then register A will be 12 after the following istruction: LD A,(BC), load register A with whatever is at the contents of register pair BC. If we use an immediate 16 bit value, say 612, rather than register pair BC and location 612 and 613 contain the address 6000, then register A will again be equal to 12 after the instruction: LD A,(612) is executed. Remember that whatever is parenthesised is assumed to contain a 16 bit address. Therefore locations 612 and 613 are assumed to hold the address of the value to be used. Any 16 bit value is stored in reverse order in memory from what it looks like as an operand. Therefore, the 16 bit immediate operand 1234 hex is stored as 3412 hex in memory.

A special case of indirect addressing is the index addressing mode. This mode uses the IX and IY registers as indices into a list of memory. The starting address of the list is given as a 16 bit immediate operand. The immedate address is added to the contents of the index register to point to a value in memory. For example, if a list starts at location 500 hex and register IX is equal to 3 then the instrucition: LD A,(IX+500) will load register A with the byte at location 500 hex + 3 = 503 hex. As you can see it is similar to a one dimensional array in BASIC.

This information should get you started. Next time we'll start writing some machine code routines. (see FIG. 15-2 for tables)

---------- F C A U G SOFTWARE ----------

E-Z EDIT - ** NEW ** An alternative tape/disk editing program that is easy to understand and use. Price $15.

E-Z COPY - The utility program that lets you copy only the blocks you need. Save time by not copying empty blocks. Make back up copies of all your vauluable software. Price $15.

FCAUG BASIC PROGRAM LIBRARY #1 - A collection of over 30 programs including many games, graphics, home management applications and sound programs. Price $10.

FCAUG BASIC PROGRAM LIBRARIES #2 & #3 - Even more programs demonstrating the infinite capabilities of the ADAM. Some really superb graphics and games programs! Price $10 each.

CP/M 2.2 COMMUNICATIONS PACKAGE - Includes MEX, ADAMBOOT and two versions of MODEM 7. These are powerful and versatile programs useful for transmitting binary files. Price $10.

PUBLIC DOMAIN PROGRAMS - ** NEW ** Quality programs that were never released commercially by Coleco and have been placed in the public domain. Included are: Troll's Tale, Super Subroc, Jeopardy, Hard Hat Mack/Pinball Construction (both on same media) and SmartBASIC V2.0. Price $10 each.

Software available on tape or disk.

First Table

FIG. 15-2 Z-80 CPU Instructions Sorted by Mnemonic

OBJ CODE	SOURCE STATEMENT	OBJ CODE	SOURCE STATEMENT	OBJ CODE	SOURCE STATEMENT
8E	ADC A, (HL)	DD29	ADD IX, IX	CB49	BIT 1, C
DD8E06	ADC A, (IX + d)	DD39	ADD IX, SP	CB4A	BIT 1, D
FD8E06	ADC A, (IY + d)	FD09	ADD IY, BC	CB4B	BIT 1, E
8F	ADC A, A	FD19	ADD IY, DE	CB4C	BIT 1, H
88	ADC A, U	FD29	ADD IY, IY	CB4D	BIT 1, L
89	ADC A, C	FD39	ADD IY, SP	CB56	BIT 2, (HL)
8A	ADC A, D	A6	AND (HL)	DDCB0556	BIT 2, (IX + d)
8B	ADC A, E	DDA605	AND (IX + d)	FDCB0556	BIT 2, (IY + d)
8C	ADC A, H	FDA605	AND (IY + d)	CB57	BIT 2, A
8D	ADC A, L	A7	AND A	CB50	BIT 2, B
CE20	ADC A, N	A0	AND B	CB51	BIT 2, C
ED4A	ADC HL, BC	A1	AND C	CB52	BIT 2, D
ED5A	ADC HL, DE	A2	AND D	CB53	BIT 2, E
ED6A	ADC HL, HL	A3	AND E	CB54	BIT 2, H
ED7A	ADC HL, SP	A4	AND H	CB55	BIT 2, L
86	ADD A, (HL)	A5	AND L	CB5E	BIT 3, (HL)
DD8605	ADD A, (IX + d)	E620	AND N	DDCB055E	BIT 3, (IX + d)
FD8605	ADD A, (IY + d)	CB46	BIT 0, (HL)	FDCB055E	BIT 3, (IY + d)
87	ADD A, A	DDCB0546	BIT 0, (IX + d)	CB5F	BIT 3, A
80	ADD A, B	FDCB0546	BIT 0, (IY + d)	CB58	BIT 3, B
81	ADD A, C	CB47	BIT 0, A	CB59	BIT 3, C
82	ADD A, D	CB40	BIT 0, B	CB5A	BIT 3, D
83	ADD A, E	CB41	BIT 0, C	CB5B	BIT 3, E
84	ADD A, H	CB42	BIT 0, D	CB5C	BIT 3, H
85	ADD A, L	CB43	BIT 0, E	CB5D	BIT 3, L
C620	ADD A, N	CB44	BIT 0, H	CB66	BIT 4, (HL)
09	ADD HL, BC	CB45	BIT 0, L	DDCB0566	BIT 4, (IX + d)
19	ADD HL, DE	CB4E	BIT 1, (HL)	FDCB0566	BIT 4, (IY + d)
29	ADD HL, HL	DDCB054E	BIT 1, (IX + d)	CB67	BIT 4, A
39	ADD HL, SP	FDCB054E	BIT 1, (IY + d)	CB60	BIT 4, B
DD09	ADD IX, BC	CB4F	BIT 1, A	CB61	BIT 4, C
DD19	ADD IX, DE	BC48	BIT 1, B	CB62	BIT 4, D

Last Table - continued from p.19

FIG. 15-2 (cont'd.)

OBJ CODE	SOURCE STATEMENT	OBJ CODE	SOURCE STATEMENT	OBJ CODE	SOURCE STATEMENT
CB22	SLA D	FDCB053E	SRL (IY + d)	93	SUB E
CB23	SLA E	CB3F	SRL A	94	SUB H
CB24	SLA H	CB38	SRL B	95	SUB L
CB25	SLA L	CB39	SRL C	D620	SUB N
CB2E	SRA (HL)	CB3A	SRL D	AE	XOR (HL)
DDCB052E	SRA (IX + d)	CB3B	SRL E	DDAE05	XOR (IX + d)
FDCB052E	SRA (IY + d)	CB3C	SRL H	FDAE05	XOR (IY + d)
CB2F	SRA A	CB3D	SRL L	AF	XOR A
CB28	SRA B	96	SUB (HL)	A8	XOR B
CB29	SRA C	DD9605	SUB (IX + d)	A9	XOR C
CB2A	SRA D	FD9605	SUB (IY + d)	AA	XOR D
CB2B	SRA E	97	SUB A	AB	XOR E
CB2C	SRA H	90	SUB B	AC	XOR H
CB2D	SRA L	91	SUB C	AD	XOR L
CB3E	SRL (HL)	92	SUB D	EE20	XOR N
DDCB053E	SRL (IX + d)				

FIG. 15-2 (cont'd.)

OBJ CODE	SOURCE STATEMENT	OBJ CODE	SOURCE STATEMENT	OBJ CODE	SOURCE STATEMENT
CB63	BIT 4, E	BF	CP A	ED48	IN C, (C)
CB64	BIT 4, H	B8	CP B	ED50	IN D, (C)
CB65	BIT 4, L	B9	CP C	ED58	IN E, (C)
CB6E	BIT 5, (HL)	BA	CP D	ED60	IN H, (C)
DDCB066E	BIT 5, (IX + d)	BB	CP E	ED68	IN L, (C)
FDCB066F	BIT 5, (IY + d)	BC	CP H	34	INC (HL)
CB6F	BIT 5, A	BD	CP L	DD3405	INC (IX + d)
CB68	BIT 5, B	FE20	CP N	FD3405	INC (IY + d)
CB69	BIT 5, C	EDA9	CPD	3C	INC A
CB6A	BIT 5, D	ED89	CPDR	04	INC B
CB6B	BIT 5, E	EDA1	CPI	03	INC BC
CB6C	BIT 5, H	EDB1	CPIR	0C	INC C
CB6D	BIT 5, L	2F	CPL	14	INC D
CB76	BIT 6, (HL)	27	DAA	13	INC DE
DDCB0676	BIT 6, (IX + d)	35	DEC (HL)	1C	INC E
FDCB0678	BIT 5, (IY + d)	DD3505	DEC (IX + d)	24	INC H
CB77	BIT 6, A	FD3505	DEC (IY + d)	23	INC HL
CB70	BIT 6, B	3D	DEC A	DD23	INC IX
CB71	BIT 8, C	05	DEC B	FD23	INC IY
CB72	BIT 6, D	08	DEC BC	2C	INC L
CB73	BIT 6, E	0D	DEC C	33	INC SP
CB74	BIT 6, H	15	DEC D	EDAA	IND
CB75	BIT 6, L	18	DEC DE	EDBA	INDR
CB7E	BIT 7, (HL)	1D	DEC E	EDA2	INI
DDCB057E	BIT 7, (IX + d)	25	DEC H	EDB2	INIR
FDCB057E	BIT 7, (IY + d)	2B	DEC HL	E9	JP (HL)
CB7F	BIT 7, A	DD2B	DEC IX	DDE9	JP (IX)
CB78	BIT 7, B	FD2B	DEC IY	FDE9	JP (IY)
CB79	BIT 7, C	2D	DEC L	DA8405	JP C, NN
CB7A	BIT 7, D	3B	DEC SP	FA8405	JP M, NN
CB7B	BIT 7, E	F3	DI	D28405	JP NC, NN
CB7C	BIT 7, H	102E	DJNZ DIS	C38405	JP NN
CB7D	BIT 7, L	FB	EI	C28405	JP NZ, NN
DC8405	CALL C, NN	E3	EX (SP), HL	F28405	JP P, NN
FC8405	CALL M, NN	DDE3	EX (SP), IX	EA8405	JP PE, NN
D48405	CALL NC, NN	FDE3	EX (SP), IY	E28405	JP PO, NN
CD8405	CALL NN	08	EX AF, AF'	CA8405	JP Z, NN
C48405	CALL NZ, NN	EB	EX DE, HL	382E	JR C, DIS
F48405	CALL P, NN	D9	EXX	182E	JR DIS
EC8405	CALL PE, NN	76	HALT	302E	JR NC, DIS
E48405	CALL PO, NN	ED46	IM 0	202E	JR NZ, DIS
CC8405	CALL Z, NN	ED56	IM 1	282E	JR Z, DIS
3F	CCF	ED5E	IM 2	02	LD (BC), A
BE	CP (HL)	ED78	IN A, (C)	12	LD (DE), A
DDBE05	CP (IX + d)	DB20	IN A, (N)	77	LD (HL), A
FDBE06	CP (IY + d)	ED40	IN B, (C)	70	LD (HL), B

FIG. 15-2 (cont'd.)

OBJ CODE	SOURCE STATEMENT	OBJ CODE	SOURCE STATEMENT	OBJ CODE	SOURCE STATEMENT
71	LD (HL), C	FD4605	LD B, (IY + d)	66	LD H, (HL)
72	LD (HL), D	47	LD B, A	DD6605	LD H, (IX + d)
73	LD (HL), E	40	LD B, B	FD6606	LD H, (IY + d)
74	LD (HL), H	41	LD B, C	67	LD H, A
75	LD (HL), L	42	LD B, D	60	LD H, B
3620	LD (HL), N	43	LD B, E	61	LD H, C
DD7706	LD (IX + d), A	44	LD B, H, NN	62	LD H, D
DD7006	LD (IX + d), B	45	LD B, L	63	LD H, E
DD7106	LD (IX + d), C	0620	LD B, N	64	LD H, H
DD7206	LD (IX + d), D	ED4B8405	LD BC, (NN)	65	LD H, L
DD7306	LD (IX + d), E	018405	LD BC, NN	2620	LDH, N
DD7406	LD (IX + d), H	4E	LD C, (HL)	2A8405	LD HL, (NN)
DD7506	LD (IX + d), L	DD4E05	LD C, (IX + d)	218405	LD HL, NN
DD360520	LD (IX + d), N	FD4E05	LD C, (IY + d)	ED47	LD I, A
FD7705	LD (IY + d), A	4F	LD C, A	DD2A8405	LD IX, (NN)
FD7005	LD (IY + d), B	48	LD C, B	DD218405	LD IX, NN
FD7105	LD (IY + d), C	49	LD C, C	FD2A8405	LD IY, (NN)
FD7205	LD (IY + d), D	4A	LD C, D	FD218405	LD IY, NN
FD7305	LD (IY + d), E	4B	LD C, E	6E	LD L, (HL)
FD7405	LD (IY + d), H	4C	LD C, H	DD6E05	LD L, (IX + d)
FD7505	LD (IY + d), L	4D	LD C, L	FD6E06	LD L, (IY + d)
FD360520	LD (IY + d), N	0E20	LD C, N	6F	LD L, A
328405	LD (NN), A	56	LD D, (HL)	68	LD L, B
ED438405	LD (NN), BC	DD5605	LD D, (IX + d)	69	LD L, C
ED538405	LD (NN), DE	FD5605	LD D, (IY + d)	6A	LD L, D
228405	LD (NN), HL	57	LD D, A	6B	LD L, E
DD228405	LD (NN), IX	50	LD D, B	6C	LD L, H
FD228405	LD (NN), IY	51	LD D, C	6D	LD L, L
ED738405	LD (NN), SP	52	LD D, D	2E20	LD L, N
0A	LD A, (BC)	53	LD D, E	ED788405	LD SP, (NN)
1A	LD A, (DE)	54	LD D, H	F9	LD SP, HL
7E	LD A, (HL)	55	LD D, L	DDF9	LD SP, IX
DD7E05	LD A, (IX + d)	1620	LD D, N	FDF9	LD SP, IY
FD7E05	LD A, (IY + d)	ED5B8405	LD DE, (NN)	318405	LD SP, NN
3A8405	LD A, (NN)	118405	LD DE, NN	EDA8	LDD
7F	LD A, A	5E	LD E, (HL)	ED88	LDDR
7B	LD A, B	DD5E05	LD E, (IX + d)	EDA0	LDI
79	LD A, C	FD5E05	LD E, (IY + d)	EDB0	LDIR
7A	LD A, D	5F	LD E, A	ED44	NEG
7B	LD A, E	58	LD E, B	00	NOP
7C	LD A, H	59	LD E, C	B6	OR (HL)
ED57	LD A, I	5A	LD E, D	DDB605	OR (IX + d)
7D	LD A, L	5B	LD E, E	FDB605	OR (IY + d)
3E20	LD A, N	5C	LD E, H	B7	OR A
46	LD B, (HL)	5D	LD E, L	B0	OR B
DD4605	LD B, (IX + d)	1E20	LD E, N	B1	OR C

FIG. 15-2 (cont'd.)

OBJ CODE	SOURCE STATEMENT	OBJ CODE	SOURCE STATEMENT	OBJ CODE	SOURCE STATEMENT
B2	OR D	CB8B	RES 1, E	CBB7	RES 6, A
B3	OR E	CB8C	RES 1, H	CBB0	RES 6, B
B4	OR H	CB8D	RES 1, L	CBB1	RES 6, C
B5	OR L	CB96	RES 2, (HL)	CBB2	RES 6, D
F620	OR N	DDCB0596	RES 2, (IX + d)	CBB3	RES 6, E
EDBB	OTDR	FDCB0596	RES 2, (IY + d)	CBB4	RES 6, H
EDB3	OTIR	CB97	RES 2, A	CBB5	RES 6, L
ED79	OUT (C), A	CB90	RES 2, B	CBBE	RES 7, (HL)
ED41	OUT (C), B	CB91	RES 2, C	DDCB05BE	RES 7, (IX + d)
ED49	OUT (C), C	CB92	RES 2, D	FDCB05BE	RES 7, (IX + d)
ED51	OUT (C), D	CB93	RES 2, E	CBBF	RES 7, A
ED59	OUT (C), E	CB94	RES 2, H	CBB8	RES 7, B
ED61	OUT (C), H	CB95	RES 2, L	CBB9	RES 7, C
ED69	OUT (C), L	CB9E	RES 3, (HL)	CBBA	RES 7, D
D320	OUT (N), A	DDCB059E	RES 3, (IX + d)	CBBB	RES 7, E
EDAB	OUTD	FDCB059E	RES 3, (IY + d)	CBBC	RES 7, H
EDA3	OUTI	CB9F	RES 3, A	CBBD	RES 7, L
F1	POP AF	CB98	RES 3, B	C9	RET
C1	POP BC	CB99	RES 3, C	D8	RET C
D1	POP DE	CB9A	RES 3, D	F8	RET M
E1	POP HL	CB9B	RES 3, E	D0	RET NC
DDE1	POP IX	CB9C	RES 3, H	C0	RET NZ
FDE1	POP IY	CB9D	RES 3, L	F0	RET P
F5	PUSH AF	CBA6	RES 4, (HL)	E8	RET PE
C5	PUSH BC	DDCB05AB	RES 4, (IX + d)	E0	RET PO
D5	PUSH DE	FDCB05AB	RES 4, (IY + d)	C8	RET Z
E5	PUSH HL	CBA7	RES 4, A	ED4D	RETI
DDE5	PUSH IX	CBA0	RES 4, B	ED45	RETN
FDE5	PUSH IY	CBA1	RES 4, C	CB16	RL (HL)
CB86	RES 0, (HL)	CBA2	RES 4, D	DDCB0516	RL (IX + d)
DDCB0586	RES 0, (IX + d)	CBA3	RES 4, E	FDCB0516	RL (IY + d)
FDCB0586	RES 0, (IY + d)	CBA4	RES 4, H	CB17	RL A
CB87	RES 0, A	CBA5	RES 4, L	CB10	RL B
CB80	RES 0, B	CBAE	RES 5, (HL)	CB11	RL C
CB81	RES 0, C	DDCB05AE	RES 5, (IX + d)	CB12	RL D
CB82	RES 0, D	FDCB05AE	RES 5, (IY + d)	CB13	RL E
CB83	RES 0, E	CBAF	RES 5, A	CB14	RL H
CB84	RES 0, H	CBA8	RES 5, B	CB15	RL L
CB85	RES 0, L	CBA9	RES 5, C	17	RLA
CB8E	RES 1, (HL)	CBAA	RES 5, D	CB06	RLC (HL)
DDCB058E	RES 1, (IX + d)	CBAB	RES 5, E	DDCB0506	RLC (IX + d)
FDCB058E	RES 1, (IY + d)	CBAC	RES 5, H	FDCB0506	RLC (IY + d)
CB8F	RES 1, A	CBAD	RES 5, L	CB07	RLC A
CB88	RES 1, B	CBB6	RES 6, (HL)	CB00	RLC B
CB89	RES 1, C	DDCB05B6	RES 6, (IX + d)	CB01	RLC C
CB8A	RES 1, D	FDCB05B6	RES 6, (IY + d)	CB02	RLC D

FIG. 15-2 (cont'd.)

OBJ CODE	SOURCE STATEMENT	OBJ CODE	SOURCE STATEMENT	OBJ CODE	SOURCE STATEMENT
CB03	RLC E	DE20	SBC A, N	CBE6	SET 4, (HL)
CB04	RLC H	ED42	SBC HL, BC	DDCB05E6	SET 4, (IX + d)
CB05	RLC L	ED52	SBC HL, DE	FDCB05E6	SET 4, (IY + d)
07	RLCA	ED62	SBC HL, HL	CBE7	SET 4, A
ED6F	RLD	ED72	SBC HL, SP	CBE0	SET 4, B
CB1E	RR (HL)	37	SCF	CBE1	SET 4, C
DDCB051E	RR (IX + d)	CBC6	SET 0, (HL)	CBE2	SET 4, D
FDCB051E	RR (IY + d)	DDCB05CS	SET 0, (IX + d)	CBE3	SET 4, E
CB1F	RR A	FDCB05C6	SET 0, (IY + d)	CBE4	SET 4, H
CB18	RR B	CBC7	SET 0, A	CBE5	SET 4, L
CB19	RR C	CBC0	SET 0, B	CBEE	SET 5, (HL)
CB1A	RR D	CBC1	SET 0, C	DDCB05EE	SET 5, (IX + d)
CB1B	RR E	CBC2	SET 0, D	FDCB05EE	SET 5, (IY + d)
CB1C	RR H	CBC3	SET 0, E	CBEF	SET 5, A
CB1D	RR L	CBC4	SET 0, H	CBE8	SET 5, B
1F	RRA	CBC5	SET 0, L	CBE9	SET 5, C
CB0E	RRC (HL)	CBCE	SET 1, (HL)	CBEA	SET 5, D
DDCB050E	RRC (IX + d)	DDCB05CE	SET 1, (IX + d)	CBEB	SET 5, E
FDCB050E	RRC (IY + d)	FDCB05CE	SET 1, (IY + d)	CBEC	SET 5, H
CB0F	RRC A	CBCF	SET 1, A	CBED	SET 5, L
CB08	RRC B	CBC8	SET 1, B	CBF6	SET 6, (HL)
CB09	RRC C	CBC9	SET 1, C	DDCB05F6	SET 6, (IX + d)
CB0A	RRC D	CBCA	SET 1, D	FDCB05F6	SET 6, (IY + d)
CB0B	RRC E	CBCB	SET 1, E	CBF7	SET 6, A
CB0C	RRC H	CBCC	SET 1, H	CBF0	SET 6, B
CB0D	RRC L	CBCD	SET 1, L	CBF1	SET 6, C
0F	RRCA	CBD6	SET 2, (HL)	CBF2	SET 6, D
ED67	RRD	DDCB05D6	SET 2, (IX + d)	CBF3	SET 6, E
C7	RST 0	FDCB05D6	SET 2, (IY + d)	CBF4	SET 6, H
D7	RST 10H	CBD7	SET 2, A	CBF5	SET 6, L
DF	RST 18H	CBD0	SET 2, B	CBFE	SET 7, (HL)
E7	RST 20H	CBD1	SET 2, C	DDCB05FE	SET 7, (IX + d)
EF	RST 28H	CBD2	SET 2, D	FDCB05FE	SET 7, (IY + d)
F7	RST 30H	CBD3	SET 2, E	CBFF	SET 7, A
FF	RST 38H	CBD4	SET 2, H	CBF8	SET 7, B
CF	RST 8	CBD5	SET 2, L	CBF9	SET 7, C
9E	SBC A, (HL)	CBD8	SET 3, B	CBFA	SET 7, D
DD9E05	SBC A, (IX + d)	CBDE	SET 3, (HL)	CBFB	SET 7, E
FD9E05	SBC A, (IY + d)	DDCB05DE	SET 3, (IX + d)	CBFC	SET 7, H
9F	SBC A, A	FDCB05DE	SET 3, (IY + d)	CBFD	SET 7, L
98	SBC A, B	CBDF	SET 3, A	CB26	SLA (HL)
99	SBC A, C	CBD9	SET 3, C	DDCB0526	SLA (IX + d)
9A	SBC A, D	CBDA	SET 3, D	FDCB0526	SLA (IY + d)
9B	SBC A, E	CBDB	SET 3, E	CB27	SLA A
9C	SBC A, H	CBDC	SET 3, H	CB20	SLA B
9D	SBC A, L	CBDD	SET 3, L	CB21	SLA C

```
 5 REM   **MOVING TRIANGLE**
10 HGR2: HCOLOR = 3
15 t = t+.0878: u = 79
20 v = 80*SIN(.5061+t)+90
25 w = 73*SIN(.8378+t)+90
30 x = 69*SIN(1.1868+t)+90
35 y = 78*SIN(1.5184+t)+90
40 a = 132: b = 139: c = 166
45 d = 90: e = 118
50 HPLOT a, u TO b, v TO c, w
55 HPLOT  TO a, u TO e, y TO c, w
60 HPLOT c, w TO b, v TO d, x
65 HPLOT  TO a, u TO e, y TO d, x
70 FOR n = 1 TO 500: NEXT:GOTO 10
```

```
 5 HGR
10 r = 7: d = 3: c = 3
15 INPUT " DEGREES (5 - 355)?"; a
20 HGR: n = 360/a: HCOLOR = c
25 a = a*(3.14159/180)
30 x1 = 100: y1 = 70
35 HPLOT x1, y1: i = 0
40 x2 = r*COS(a*i)+x1
45 y2 = -r*SIN(a*i)+y1
50 IF x2 < 0 OR x2 > 290 THEN 10
55 IF y2 < 0 OR y2 > 150 THEN 10
60 HPLOT TO x2, y2
65 x1 = x2: y1 = y2
70 r = r+d: i = i+1: GOTO 40
```

```
10 REM   **CROSS**
15 GR: FOR n = 1 TO 500
20 IF c > 14 THEN d = 0
25 IF c < 2 THEN d = 1
30 IF d = 0 THEN c = c-1
35 IF d = 1 THEN c = c+1
40 COLOR = c
45 HLIN 0, 39 AT x
50 VLIN 0, 39 AT y
55 IF x = 39 THEN x = 0
60 IF y = 39 THEN y = 0
65 x = x+1: y = y+1: NEXT
```

```
10 REM   **SINEWAVES**
15 HGR: FOR r = 2 TO 79 STEP 5
20 tpi = 6.28318
25 FOR i = 0 TO 664 STEP 7
30 x = i/2.58
35 y = r*SIN(i/360*tpi)
40 HCOLOR = 9
45 HPLOT x, 80-y
50 z = r*COS(i/360*tpi)
55 HCOLOR = 8
60 HPLOT x, 80-z
65 NEXT: NEXT: GET a$: TEXT
```

```
 10 REM  **Gregorian Calendar**
 15 HOME
 20 DIM c$(42), d$(31), e(12)
 25 FOR i = 1 TO 31: READ d$(i): NEXT i
 30 FOR i = 1 TO 12: READ e(i): NEXT i
 35 DATA "  1","  2","  3","  4","  5","  6","  7","  8","  9"," 10"
 40 DATA " 11"," 12"," 13"," 14"," 15"," 16"," 17"," 18"," 19"," 20"
 45 DATA " 21"," 22"," 23"," 24"," 25"," 26"," 27"," 28"," 29"," 30"," 31"
 50 DATA   31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
 55 PRINT: PRINT "  INPUT MONTH AND YEAR"
 60 PRINT: INPUT "  (i.e. MM,YY) =>"; m, y
 65 IF y < 100 THEN  y = y+1900
 70 PRINT: PRINT " SU MO TU WE TH FR SA": PRINT
 75 j = 367*y-INT(7*(y+INT((m+9)/12))/4)+INT(275*m/9)+1721031
 80 k = 0: IF m <= 2 THEN  k = -1
 85 j = j-INT(3*(INT((y+k)/100)+1)/4)
 90 k = e(m): IF m <> 2 THEN 115
 95 w = INT(y-100*INT(y/100)): x = INT(y-4*INT(y/4))
100 z = INT(y-400*INT(y/400))
105 IF x <> 0 THEN 115
110 IF w = 0 AND z <> 0 THEN 115: k = 29
115 x = j-7*INT(j/7)
120 FOR i = 1 TO 42: c$(i) = "   ": NEXT i
125 FOR i = 1 TO k: c$(i+x) = d$(i): NEXT i
130 FOR i = 1 TO 6: j = 7*i
135 PRINT c$(j-6); c$(j-5); c$(j-4); c$(j-3); c$(j-2); c$(j-1); c$(j)
140 NEXT i
145 PRINT: INPUT "  ANOTHER....(y/n)? "; a$: IF a$ = "y" THEN 55
```