WISCONSIN COMPUTER SOCIETY
NEWSLETTER

****

Volume #2, Issue #9 October 1977 Don Stevens, Editor

****

MEETING NOTICE

Our meeting will be held at 1:00 p.m., Saturday, October 1, 1977, at the Waukesha Technical Institute (room 202 - Administration Bldg.)

PROGRAM AGENDA

WCTI will give us a tour of their Computer facilities which is used for Administration and Student purposes. Some of the components in their computer system are Burroughs B-6700 Computer, 1.2 megabyte core, 500 megabyte disc, 2 card readers, 2 printers, and 28 data communication ports.

****

LOOKING FOR AN EXCELLENT BUY ON A ASR-33 TELETYPE complete with Modem? Contact the writer at once. Only one (1) left.

****

FREE - FREE copies of COMPUTER NOTES from MITS available at the meeting. Limited supply.

****

MOHR LABS, INC. of Madison will be our guest speakers for our NOVEMBER Meeting. They have been involved in microcomputer system design for a number of years, and they will be demonstrating their BJ-4 Desk Top Computer System. More details next issue on this.

****

Micro-Com, Inc., P.O. Box 4069, Pompano Beach, Florida, 33063, offers TDL Software packages for the Z-80 systems such ad Digital Group.

****

NEWSLETTER INFO

Please send your info for the Newsletter to:

Don Stevens

Don Stevens, Editor
Wisconsin Computer Society
P.O. Box 159
Sheboygan Falls, Wisc. 53085

Hexadecimal Program Tape Format

:100C00002100OD232206F82B237EFEFFC2080C23B1

Have you often wondered what the above print out meant?

This is a hexadecimal tape format used primarily for reading in and punching out paper tape. It is arrayed as a memory image and blocked into discrete records. Each record contains the record length, record type, memory address, and checksum information in addition to data. A frame by frame description is as follows:

Frame 0	Record Mark. Signals the start of a record. The ASCII character colon (":"Hex 3A) is used as the record mark.
Frames 1,2 (0-9, A-F)	Record Length. Two ASCII characters representing a hexadecimal number in the range 0 to 'FF'H(0 to 255). This is the count of the actual data bytes in the record type or checksum. A record length of 0 indicates end of file.
Frames 3 to 6	Load Address. Four ASCII characters that represent the initial memory location where the data following will be loaded. The first byte is stored in the location pointed to by the load address,succeding data bytes are loaded into ascending addresses.
Frames 7,8	Record Type. Two ASCII characters. Currently all record are type 0, this field is reserved for future expansion.
Frames 9 to 9 2 (Record length) -1	Data. Each 8-bit memory word is represented by two frames containing the ASCII character (0-9, A-F) to represent a hexadecimal value 0 to 'FF'H (0 to 255).
Frames 9 2 (Record Length) to Frames 9 2 (Record Length) 1	Checksum. The checksum is the negative of the sum of all 8-bit bytes in the record since the record mark (":") evaluated modulus 256. That is, if you add together all the 8-bit bytes, ignoring all carries out of the 8-bit sum, then add the checksum, the result is zero.

"Hexadecimal Program Tape Format",by Ed Gerri appears courtesy the NNJACC Newsletter, July 1977. Ed Gerri is an ACC-NJ member.

PIXMOXING

The Processor Technology Video Display Module(VDM) and MITS BASIC make a great combination. BASIC's PEEK (look at memory) and POKE (store in memory) open a wealth of video games and simulations which are unavailable to 'Teletype replacement' video displays or BASICs with no PEEK or POKE.

PIXMOX is a program which takes a picture you have drawn in DATA statements in the program, and distributes the pieces of the picture randomly about the VDM screen.

Then, the VDM 'Moon Man' (the bell character) is given the task to take the strewn pieces, and rebuild the picture. He does so in a series of steps:

1. He moves to the place on the VDM where the first picture character is to go. When he arrives, he 'blinks' himself (the program pokes a 128 which is the reverse video cursor). If there is already a character there, he moves it out of the way.
2. He then finds the proper picture character which belongs in that location and blinks it. This is not necessarily the nearest one, but rather any random character which is not already in place, and which is the character needed.
3. He then crawls under the character (i.e. you no longer see the Moon Man, but rather the character 'on his shoulders').
4. He then moves this character into position, and goes back to step 1, for the next picture element.

The challenge in programming PIXMOX was the fact that there are other characters in the way while Moon Man is moving. Several techniques are used to get around the obstacle. The subroutine at 8000 tries to go to the right then left, or left then right to get by. If that is unsucessful, control transfers to 9000 which is a general random move routine. This routine picks a random DY and DX (direction Y and direction X) and a random length L to move. After moving, it sometimes tries moving right or left and returns, and other times goes back to 9000. This causes the Moon Man to take on a rather erratic walk, but is necessary logic to help him escape when nearly trapped by a group of characters on the screen?

If you have some suggestions about making the Moon Man smarter, I would be interested in trying them. The more quickly and directly he accomplishes his task,the more interesting it is to watch the program.

Here is a little about the ideas used in the program:

The DEF statements at the beginning of the program are used to generate addresses of locations on the VDM. For example, DEF FNP(Z)= VB + 64 * Y + X is used to calculate the address of the character at location Y,X on the VDM. VB is the start of the VDM (7000 Hex). (Note the VDM must be under 8000 hex to be POKEd from early versions of MITS BASIC - - See P.S. on modifying 8K v3.1). The 64 is used because to go 1 row in the Y direction, you have to go 64 characters down. Thus, 0,0 is at the top left corner of the screen and 15,63 is at the bottom right.

The statements using the SGN function are used to determine the direction from where the Moon Man is (X,Y) to some goal, such as the location where a block of the picture is to be built, or the location where a block now is that has to be moved into place. For example if the Moon Man is at 5,34 and the place he wants to (say GY,GX) is 3,50 then SGN(GY-Y) will produce the SGN of 3-5 which is the SGN of -2, which is -1. Thus, using the SGN function the Y and X direction are determined, as -1, 0, or 1. This causes the Moon Man to move 1 square at a time toward the goal.

When moving, the Moon Man usually appears to move on a diagonal, then horizontally. This is because of the width (64) versus height (16) of the VDM, and the fact that he moves 1 X and 1 Y until he is in the right row or column. It would have been possible to calculate a direction for him to move which would go directly to the goal, but my reason for not going with fractional X and Y was so that the program could be hand converted to assembler and not have any floating point arithmetic to take care of.

HAVE FUN

P.S. This program will run on any 8K or larger version of MITS BASIC. NOTE you must be able to POKE your VDM for this program to work. This means having it addressed below 8000 Hex as mentioned above, having a version which allows PEEK and POKE to work over 8000 Hex, or means of modifying BASIC to allow PEEKing or POKEing above 8000 Hex. If you are running 8K version 3.1, you can defeat the negative number chacking in PEEK and POKE by NOPing the 3 byte instruction 'JM 'CERR' at 646 Hex. The address for 8K v 3.2 is 649 Hex. Some versions of 8K v 3.1 are 1 byte off from this address, so make sure the instruction is a 'JM' (FA Hex), and patch a 00 into it and the next two bytes. With this modification, you will now be able to PEEK and POKE using a negative value for addresses above 32K. If your VDM is not at 7000 Hex, change the 'VB = 28672' instruction in the program to your VDM beginning memory address. For example, if your VDM is at CC00 Hex,this is -13312 in decimal. (F000 is -4K (K = 1024), E000 is -8K, D000 is -12K, CC00 is -13K or -(13*1024) = -13312.

HAPPY PIXMOXING..... Ward Christensen

"PIXMOXING" by Ward Christensen appears courtesy the Chicago Area Computer Hobbyist's Exchange newsletter, the CACHE Register, Vol. 2 No. 5, August 1977.