
The
Commodore
User
Group of Sask.'s
Monitor
MARCH
1987

OBLIGATORY STUFF

PRESIDENT	Richard Maze
VICE-PRESIDENT	Ed Dietrich
SECRETARY-TREASURER	Gordon Glew
LIBRARIAN	Earl Brown
ASS'T LIBRARIAN	Randy Sloboda
EDITOR	Ken Danylczuk
ASS'T EDITOR	Greg Rezansoff
MEMBERS AT LARGE	Steve Bogues
	Harry Chong

THE MONITOR is published monthly by the COMMODORE USERS' GROUP OF SASKATCHEWAN (CUGS), Regina, Sask., Canada. It is distributed at each monthly CUGS meeting, held at 7 pm, the FIRST WEDNESDAY of every month. Meetings are usually held in the North-West Leisure Centre, at the corner of Rochdale Boulevard and Arnason St., Regina.

Anyone interested in computing, especially on the C64, 128 or 64C, is welcome to attend any meeting. Out of town members are welcome, but may be charged a sml1 mailing fee for newsletters. Members are welcome to submit public domain software for inclusion in the CUGS DISK LIBRARY. Any member may purchase disks from the club library for a nominal fee. The club library looks for programs listed in such magazines as COMPUTE, GAZETTE, RUN, AHOY, TPUG, COMMODORE COMPUTING, etc. these programs are made available to members who purchase the magazines.

IN THIS ISSUE:

MARCH MAZE	- * W.O.W. from our prez!
MEETING PLACE	- Date, Time, Place and Agenda
DE-BEGGING	- Sharing the wealth
ZIPPERS	- Quick tricks for programs.
RICHARD'S BASIC	- Learning about inputs.
DISK-ETIQUETTE	- Library comments by Earl.
SPEAKING IN TONGUES	- Editorial on languages
ML ED-IFICATION	- co-info. on Ed.'s ML offerings

(* = Words Of Wisdom)

EDITORIAL

EDITORIAL

For most of us (C.N.= computer nuts) learning to progrm meant learning the BASIC language. Lately, magazines, educational institutes and publications, even computer salespeople suggest we might have seriously damaged our entire computing life by not learning some OTHER language as well as (or instead of) BASIC. Most hobbyists, programming merrily in BASIC, when approached by some swaggering hi-techie who asks what our favourite programming language is, develop a twitch, stutter, stammer and make incredible long apologies for not yet mastering Pascal, LOGO, AP/L, FORTRAN, FORTH, LISP, COBOL, COMAL, "C" or Assembler! MOST would as soon be guillotined as admit to programming in BASIC, and not even structured BASIC, maybe even BASIC 2.0!!! (Shame - for shame!!!)

What is it with all these languages, and should I (you) really know more than one??

For an answer to that burning question read on! Learn the complete, unadorned, unbiased, and slightly over-simplified TRUTH about computer languages and their impact on your computing life.

This issue, we'll establish exactly what is meant by "computer language", because understanding what a language IS, will help you decide your NEED TO KNOW one! (One comment for computing purists - what you're about to read might make you shudder in its over-simplicity - but, if it bothers you, I promise to print each and every submitted criticism or comment!)

A computer doesn't understand ANY language but electrical on and off switching. It is little more than an intricate collection of on/off switches - some pre-set (ROM), some setable by the user (RAM) - and delicate electrical pathways along which travel from 8 to 32 BITS (on or offs) at a time. (APPLES, early ATARIS, TI-99S, AND MOST COMMODORES send and receive 8 BITS at a time; ATARI ST, AMIGA, IBM and MACs send and receive 16 BITS at a time.) This means we C64/C128 users use 8-bit machines - machines that "understand" (receive and interpret) 8 electrical bits (called a BYTE) at any given time. This is the machine's "native tongue", its TRUE language. Anyone following Ed's ML sessions has guessed that this language is difficult and cumbersome for humans. MOST HUMANS NEED HELP TO TALK EASILY WITH THEIR COMPUTERS!

Enter the "languages". Ingenious man decided he could use the machine to act as "interpreter" between what he said and what the machine needed to hear. This "interpreter" spoke a "higher-level language" - "higher" = above and beyond the machine's "native" tongue. Two types of "in-between" languages evolved - compiled and interpretive. The first was called an ASSEMBLER (ML compiler) which permitted the programmer to use in MNEMONICS (see Ed's ML article in this issue), which were then COMPILED into the machine's native 8-bit language. Once compiled the compiled program could be loaded [illegible]

The other type of language is rarer with one exception - BASIC is an INTERPRETED language. This means that an BASIC instruction given by a programmer is examined by the BASIC INTERPRETER each time it is given, and "translated" into appropriate 8-bit lingo for the machine to act on! If this sounds slow - you're right! EVERY LANGUAGE BEYOND TRUE MACHINE CODE INVOLVES SOME LOSS OF CONVENIENCE OR SPEED (OR BOTH) as a sacrifice for easier HUMAN use.

Of all the languages mentioned above, BASIC is the slowest but easiest for the average user. Why? Because it was deliberately created for use by ANYONE! The primary goal behind the development of BASIC was to allow any person with a need to communicate easily with the mighty computer!! The price paid was speed of execution.

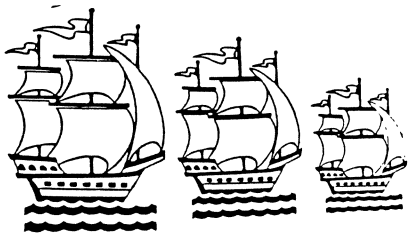
So what about all these other languages? Remember - the underlying purpose to BASIC was ease of use by ANYONE. Each of the languages listed above was created to provide easier access to DIFFERENT machine functions more for specific groups of users! Meaning what? If you fit one of the "specialized" user categories, you'll find the associated language DOES help you program. If you're NOT a specialist user, and BASIC's speed isn't a problem with you, then don't bother! Consider EXTENSIONS to BASIC which make it even easier to use, but, unless you're the kind who likes the idea of learning to speak Bulgarian for fun, don't worry about the other "tongues"!!

To summarize:

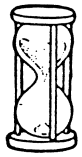
- computers "talk" in 8 to 32 bit electrical "words"
- on/off switches is their "native" tongue.
- any other language is structured for the sake of HUMAN ease of use or understanding.
- languages mean more inconvenience or time delay sacrificed for the sake of HUMAN ease of use.
- BASIC was deliberately created for use by EVERYBODY.
- other languages serve the purposes of particular "interest" groups in computer circles; if you fit one of those groups, the associated language will probably be an asset - otherwise, it's simply an academic exercise.

NEXT ISSUE: So who needs what language (and why)?

KEEP KOMPUTING!



MARCH MAZE



COMPUTERFEST 1 is over and I would like to take this opportunity to thank club members who kindly gave their time and energy to help make our club's presentation successful. Our main objective in setting up a display at COMPUTERFEST was to build awareness of CUGS. Judging by the number of people we talked with and explained the operation of our club to, we easily met this objective.

Special thanks to Ed and Ken for the programs which we had running on the two 64's. These programs brought many people over for a closer look and to talk with us. Also, thank you, Steve, for the excellent CUGS banner which marked our location clearly and attracted many people to our display.

I would also like to thank the people who so gave of their time to man the booth and answer visitors questions. Special thanks to Ed, Gordon, Harry and Ken who were present most of the day. Thank you, too, to the many others who dropped by and helped out for a period of time. I apologize for not mentioning each by names, but trying that I might inadvertantly miss a name. Thanks to all the club members who dropped by to say hello. Your comments and support were appreciated.

I especially want to thank the members of the public who dropped by and got information from us. Those who have joined our club have special meaning to the rest of us because you are the reason we set up our booth to build interest in our computer club.

A final thanks to the Apple II user's group for sponsoring COMPUTERFEST and giving us the opportunity to be a part of this great display. Judging by the crowds, there's a definite need for such a show and it was very well received.

SUPER-ZIPPER

SETTING COLORS IN MULTIPLAN

Multiplan does not allow you to change colors in the program. You can make a small modification to the loader program to set the colors to what you want. Following is the process to permanently set HesWare's Multiplan to the colors you want. (I don't know if the Epyx version loads the same or not - if it does, this process will work for it as well.

Load and list the program "mp".
You will see the following two lines:

```
1 ifothensys114958
2 o=1:load"mp.",8,1
```

Move the cursor up to the 2 and enter: 3 and press RETURN.
Enter the following line:

```
2 poke53280,bc:poke53281,sc:poke646,cc
```

NOTE: (bc is border color 0 - 15) I use 15
(sc is screen color 0 - 15) I use 15
(cc is cursor color 0 - 15) I use 0

Now scratch "mp" from your disk and save the new version of "mp" that you have just made.

```
OPEN15,8,15,"S0:MP"
CLOSE15
SAVE"MP",8:VERIFY"*",8
```

Whenever you load Multiplan the colors will be set automatically to those requested.

Disk-etiquette

The CUGS Software Library is undergoing a face lift, one we were hoping to do ages ago, but the task seemed enormous. Fortunately, the job is being shared by the club executive, and the first stage has been undertaken. It may take 'til fall to complete the transformation but most will welcome the "new look". The only bad side effect I can imagine is the deletion of magazine programs one is not entitled to.

Last month we issued the Gazette program disk #21 and the 1986 Income Tax program disk #99. The tax disk is similar to last year's but with all the necessary changes for this year. Two or three variables have been given a new identity to allow the program to run on Basic 2 as well as Basic 7; allowing the advantage of the numeric keypad for C128 owners. A small disadvantage exists to people without printers on the General program. Both the T1C Sask. tax form and the Schedule 1 Fed. tax form are too long to list on one monitor screen and scroll. If you must fill in your return from the monitor screen it may mean pressing your choice more than once, perhaps even using the CONTROL KEY to slow things down. It takes a bit more work to split the screen and I decided not to do it this year because next year's return is supposed to have some radical changes. Seems to me I've heard that one before...Oh well!!!! GAZETTE DISK #21 has all the programs from the Nov. and Dec.,1986 issues as well as Jan., 1987. If the April issue reaches us in time, disk #22 will be available with Feb, Mar, and Apr on it, otherwise next month.

In C128 mode you can load and run a basic program by typing:

```
RUN "PGM NAME"
```

On the 64 by typing:

```
LOAD"PGM NAME",8:[shifted run/stop]
```

You must have the screen clear below this entry, however, and don't forget the [:] after ,8 and before pressing run.

SEE YOU NEXT MONTH

MEETING PLACE:

AGENDA:

First a word from our sponsor ... new Business from Richard

Gordon Glew and the mighty INPUT statement

Ed ("Maddog") Dietrich and part III of ML programming

***** Coffee *** Visits *** Library Checking *******

Major Presentation - Paperclip - a Professional Word-processing program (by Richard Maze)

ML ED-ucation

[Editor's foreword: Ed had intended for a short article in the MONITOR to accompany each of his ML talks at our meetings, but time got in the way. He's caught up on himself now, and will be supplying a short article for the MONITOR designed to help and enhance his meeting presentations. In order to "catch up", we've printed the first TWO installments in this issue.]

The programs that follow are to be entered with the machine language monitor on the U3 disk used during the ml demonstrations at the meeting. Use the simple assembler (.A command) and follow the instruction sheet that was handed out previously. If you require an instruction sheet, see the Vice-president. I strongly recommend reading the areas concerning the assembler, disassembler and saving programs to disk.

The location or address of these programs will start at $C000 (hex) or 49152 (decimal). The first assembly command typed into the machine language monitor should then be .A C000 'INSTRUCTION'. The simple assembler will then supply the next memory location automatically.

```
Program 1          Program 2          Program 3

LDA #$07           LDX #$04           LDY #$02
STA $D021          STX $D020          STY $0286
RTS                RTS                RTS
```

After typing in the program, it should be checked for errors. Use the disassembly command as follows.

```
.D C000 C005
```

The program you just entered should be listed to the screen. It can also be saved to disk with the following command.

```
.S "0:NAME",08,C000,C006
```

Now comes the fun part. Exit the monitor by entering '.X' and press return. You should now be back in basic. To run the program type 'SYS 49152' and press return. The results should be quick.

PROGRAM 1 loads the accumulator with the numeric value seven, copies the contents of the accumulator into location $D021 and ReTurns from the ml Subroutine (in this case, back to basic).

PROGRAM 2 loads the X-register with the numeric value four, copies the contents of the X-register into location $D020 and ReTurns from the ml Subroutine

PROGRAM 3 loads the Y-register with the numeric value three, copies the contents of the Y-register into location $0286 and ReTurns from the ml Subroutine.

Program 4

```
LDA #$07
LDY #$04
LDX #$02
STA $D021
STX $D020
STY $0286
RTS
```

PROGRAM 4 can be disassembled with the command '.D C000 C00F' or saved with the command:

```
'.S "0:NAME",08,C000,C010'.
```

If you have entered and run programs one through three, you should be able to guess the results of program 4. You have learned to access three 'control' locations in ml using all three data handling registers on the 6510 microprocessor.

```
$D021 - 53281  background color
$D020 - 53280  border color
$0286 - 646    character color
```

If you have entered and run any of the above programs, you have successfully taken direct command of your computer and forced it to do your bidding without the coddling safety net of the basic interpreter. Best of all, it wasn't even that hard, was it? If however, you are unable to enter the programs correctly or have any other problems, do not hesitate to ask questions.

Sir Richard's BASIC

A BASIC AUTOMATIC LOADER

It is sometimes necessary to have one program load and run another program. One direct application is to build a loader program which is used to select a program from a menu. This loader program should be the first one on a disk, and is loaded with: load "*",8 , and when run automatically loads and runs the program selected.

The list can be obtained in many ways - from data statements in the loader, or built into a sequential file are two methods.

The routine involves printing the loading and starting message on the screen and using dynamic keyboard technique to press RETURN over each line.

To use this routine, two things must be known:

1) the name of the program;
2) the starting instruction (load or sysxxxxx).

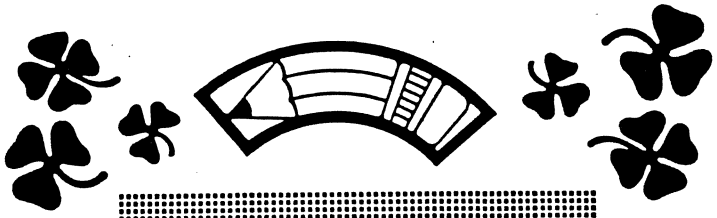
I will assume the name of the program is stored in the variable a$ and the starting instruction is in b$. The routine is then:

```
300 c$=chr$(34):rem quote
310 print "<HOME><CD><CD><CD>load"c$a$c$",8,1"
320 print "<CD><CD><CD><CD><CD>"b$"<HOME>"
330 poke631,13:poke632,13:poke198,2:end
```

Line 310 positions the load message on the screen. Use the <HOME> instead of clearing the screen and you can print a load message on the screen as well. Line 320 displays the "run" or "sysxxxxx" message. Make sure you have the proper number of cursor down's or the RETURN will miss the message. The <HOME> at the end of this line puts the cursor back to the proper position to allow the next line to execute the message. Line 330 puts 2 carriage returns [chr$(13)] in the keyboard buffer. These RETURNs should fall on the line printed on the screen and cause them to be executed.

The only thing you must be careful about is to make sure that you do not have anything else printed on these screen lines or a SYNTAX ERROR will occur and the programs will not load properly. A good way to prevent this from happening is to clear the screen and put a "loading - please wait" message in the middle of the screen (line 13) just before entering this routine.

If you want to get fancy, change the cursor color to the color of the screen (in the string in line 310) and the loader will work without the commands being visible.



W A N T E D - H E L P !

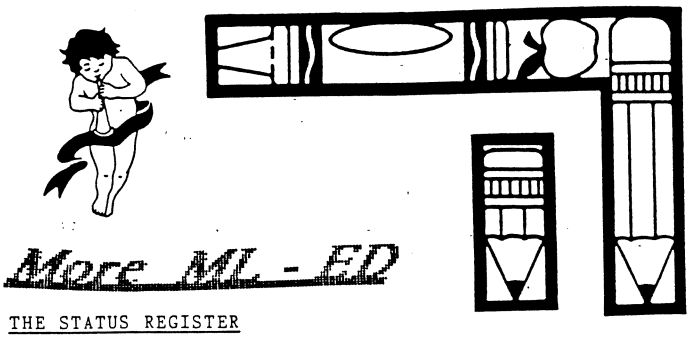
Anyone with knowledge

of PASCAL compilers

available for the C64.

Please contact:

Richard Maze 586-3291



THE STATUS REGISTER

This register is the heart of the decision-making process in all BASIC or machine language programs. Many of the instruction set commands 'condition' certain bits of this register leaving a trace of what data has been most recently processed. Using the 'compare' and 'branch' set of instructions to test the individual bits of the status register will enable the program to make decisions.

The individual bits of the status register are commonly known as flags. These flags are:

BIT#	7	6	5	4	3	2	1	0
FLAG	N	V	-	B	D	I	Z	C
	e	o e		r	e	n	e	a
	g	r		e	c	t	r	r
	a	f		a	i	e	o	r
	t	l		k	m	r		y
	i	o			a	r		
	v	w			l	u		
	e					p		
						t		

BIT 7 N (Negative)- This comes from the fact that the 6502/6510 considers any number greater than 127 as NEGATIVE. If the data in the active register has the high bit set ($80 or 128), the data is greater than 127 and the N flag will be set to 1. If the data in the active register does not have the high bit set (less than 128 or $80), the N flag will be cleared (0). Two common commands which check this flag are BPL (Branch if PLus) and BMI (Branch if MInus). This flag is set/reset by any instruction which affects any memory register (including the accumulator and index registers).

BIT 6 V (oVerflow)- This is short for Signed Arithmetic Overflow. This flag is rarely used by machine language programmers since it is only conditioned by addition and subtraction commands and is meaningful only if the numbers concerned are 'signed' (carrying a positive or negative sign). Look it up yourself. Also look up 'two's complement' form of numbering. This will give an explanation of 'negative' numbers. The V flag can also be set by hardware thus programmers can check the input/output ports of interface adapter chips. The flag may be cleared with the instruction CLV (CLear oVerflow). The commands checking this flag are BVS (Branch if oVerflow Set) or BVC (Branch if oVerflow Clear).

BIT 5 - unused

BIT 4 B (Break)- Break Indicator. This flag is set by the BRK (break) instruction. This informs the microprocessor of the type of interrupt occurring.

BIT 3 D (Decimal)- Decimal mode indicator. Instructions are SED (SEt Decimal mode) and CLD (CLear Decimal mode). Setting this flag causes the microprocessor to add and subtract in 'Binary Coded Decimal' mode. I am told that this makes the 6510 a very powerful chip but as this mode is neither true binary or true decimal these commands are noted only by their absence in my programs.

BIT 2 I (Interrupt)- Interrupt disable. This flag may be set with the instruction SEI (SEt Interrupt) and cleared with CLI (CLear Interrupt). Setting this flag inhibits further interrupts of the microprocessor (key-scan, timer update, etc.).

BIT 1 Z (Zero)- Any of the instructions that affect the accumulator or index registers will affect this flag. Like all increment and decrement instructions affecting any memory location. If the memory register affected contains the value zero the Z flag will be set (1), if the data is non-zero the flag will be cleared (0). This flag is also conditioned by the CMP,CPY,CPX (CoMPare) instructions. If the compare instruction succeeds (is equal) the Z flag is set (1). [e.g. - CPX #$08, if the X-register contains the numeric value 8, the Z flag will be set (1), if the X-register contains any other value the Z flag will be cleared (0).] The commands testing the Z flag are BEQ (Branch if EQual), BNE (Branch if Not Equal).

BIT 0 C (Carry)- This flag is set (1) by the SEC (SEt Carry) command and cleared (0) with the CLC (CLear Carry) command. It is also used as a 'carry' when using the addition and subtraction commands. It is a very useful flag when using the compare commands (CMP,CPX,CPY). [e.g. - CPY #$08 - if the Y register contains a number equal to or greater than 8, the carry flag will be set (1). If the Y-register contains a number less than 8, the carry flag will be cleared (0).] The commands testing the C flag are BCC (Branch if Carry Clear) and BCS (Branch if Carry Set.

This article, while thoroughly confusing, is by no means exhaustive. There are many commands I have not mentioned which condition or test these flags. The flags tested or conditioned by each command are shown in the instruction set in machine language manuals.

In an aside for the intimidated, I should mention that I was programming for some time, setting and testing two flags in my programs before I discovered what a Status Register was and that there were four more flags available for my use.

Here then is my humble if ignorant opinion of the usefulness of the various flags of the status register:

N - used often, usually for loops

V - used rarely, only by those involved in 1120 digit precision or by those modifying DOS (e.g. fast loaders, fast format)

B - used by the computer, not the programmer

D - A very powerful flag I never use. Having grown up with decimal, being brainburnt by binary and mentally skewed by hexadecimal, I drew the line at Binary Coded Decimal.

I - rarely used, but essential in certain situations.

Z - used very often, usually for loops

C - used most often, primarily for program testing and branching, also for addition and subtraction.

Original Offer

BASIC BEGGING!

Being editor of a prestigeous monthly like the MONITOR is a lot like driving a car with a leaky gas tank! As you stand in the middle of the Lewvan Express(?)way trying desparately to flag down a friendly driver, you think how sure you were that the tank wasn't that low when you left home.

That's a long-winded way of starting my usual monthly "beg" for MONITOR material! Although I have a couple of regular monthly contributors, I KNOW MORE OF YOU ARE USING YOUR COMPUTERS! We need: a variety of product reviews of new (and not so new) software and hardware, someone to do an article or five on beginning communication (modem hows and why's), MANY more ZIPPERS, jokes, cartoons, artwork, programs, comments, criticisms, bouquets, ... you name it! I OFFER YOU THE OPPORTUNITY to try your stuff out on a local club before you try to sell it to COMPUTE! or AHOY! Offerings in English or French, on disk or written on paper with a supermarket pencil are ALL accepted!

QUALITY

New CUGS Disk Library Items

CUGS JUKEBOX #1 #90

JUKEBOX	20
BRK MY STRIDE	27
KARMA CHAMELEON	36
MANIAC	32
RUNNIN'	39
TELEPHONE	23
UPTOWN GIRL	34
THE THUNDERER	38
MINUTE WALTZ	34
THOSE/DAYS	21
BILL BAILEY	20
MAPLELEAF RAG	21
FUR ELISE	22
SHE WORKS HARD	20
ENTERTAINER	19
MUSKRAT RAMBLE	21
CALIF. GIRLS	18
STAR WARS	32
MR. ROBOTO	40
THE GODFATHER	19
STAR TREK THEME	20
SURFING USA	21
MEMORIES	39
SUPERMAN 1	44

CUGS JUKEBOX#2 #91

JUKEBOX	20
HAPPY SONGS	31
E.T. THEME	27
HARVEST HOME	15
11 RHAP. IN BLUE	39
BEAT IT	39
BILLIE JEAN	43
BRANDENBURG1	38
BRANDENBURG2	21
JOPLIN RAG	37
GHOSTBUSTERS	21
HILL ST. BLUES	23
WILLIAM TELL	34
WALTZ-BEETHOVEN	24
ISLES IN STREAM	26
HUNGARIAN RHAP.	20
ROCKY 3 THEME	24
STAIRWAY	44
THIS IS IT!	23
BORN TO RUN	24
TWIST OF FATE	34
FLASHDANCE	28
BETH	20

CUGS SASK. TAX #99

-programs for preparing and printing the 1986 income tax for Sask. Residents, by our own librarian and tax guru EARL BROWN.

CUGS GAZETTE #21

3.<---------->	1
1526 UNDERLINER	5
SPEEDSCRIPT 3.2	25
TURBO FORMAT	7
BUMP 'N RUN	2
BUMP'N RUN.49152	7
EXAMINER.BOOT	2
BASIC EXAMINER	10
POLAR ART	11
POLAR ART/128	6
BACH MINUET/128	8
MULTITASKER.BOOT	2
MULTITASKER	4
DRAW/128	8
OBSTACLE/128	6
MATCH BLOX	14
MATCH BLOX/4/16	14
FILL.BOOT	2
FILL-64	21
FILL/DEMO	11
FILL/PLAYER	2
KEYWORDS/128	9
4.<---------->	1
MOON RESCUE/128	13
SPRITE LOCATER	8
SPRITE GRAPH	4
PEGS/PEGS-VIC	9
Q-BIRD	18
ANIMAL SHOW/128	74
QUICKSORT/128	5
QUICKSORT DEMO/128	3
BAR CHARTER	7
FAST DUMP/128/64	6
VIDEO SETUP	11
MIS-MATCHER	9
MIS-MATCHER/4/16	9
CUSTOM ENV/128	5
VF XVI/128	9
AUTO RACE/128	3
CARSHAPE	1
TILES/128	4
5.<->	1
MEDIUM RES.BOOT	1
MEDIUM RES	2
MEDIUM RES DEMO	5
VIDEO SETUP/128	15
FUNCTION KEYS	2
FKEYS.49152	2
CONNECT 'EM	21
CONNECT 'EM/128	19
INFO GEN	8
INFO PLEASE	2
INFO.49152	2
ICON CHANGER	13
KEYWORD CONSTRUCT	7
DATA-AID	2
SAINTS/128	7
JOY/128	21
DECIPEDE	8
PROOFREADER	6
MLX	18

TPUG C64 MARCH #M1

LIST-ME	11
MASH.C	31
ELEC SRVC CALC.C	33
MAG INDEX.C	34
DONKEY DONG.C	67
CLUB MAIL LIST.C	40
MOMO BOOT.C	2
MOMO PICTURE.D	33
MOMO SET.D	7
MOMO PRINT.D	11
MATH.C	18
BOOT VALLEY V2.C	1
CHARSET VALLEY.D	9
VALLEY BASIC.D	72
BRADLEY	1
SHEVLIN	1
DOW	1
KARNAK	1
IDEAL MASS.Z	34
STARS BAS PR.Z	38
FRENCH VERBS.Z	97
BASIC AID INST.C	47
BASIC AID.C	21
FILE COPY.C	19
SD COPY/ALL.C	14

TPUG MORE MARCH #M2

LIST-ME	11
BASEBALL INST.C	12
BASEBALL.C	44
BASEBALL DATA.D	4
DISK DOCTOR.C	24
PIC LOADER.C	8
COLOURS.D	28
TITLE.D	32
MARS.D	32
BIPLANE.D	32
SHIP.D	32
LANDSCAPE.D	32
AUTO.D	32
GIRL.D	32
LIST-ME INVADE.L	4
INVADERS.C	25
HORSE RACING.C	32
DODGE CARS.C	28
SHOOTOUT RULES.C	7
SHOOTOUT.C	22
R2DIVISION	101
COUNT 1-8.C	32
C64 DT.C	10
DISKALC.C	27

Will install
Reset
switch
on C64 - $10
comes with un-new
program
Call Barry Bircher
(359-1925)

Meeting: March 4,
N.W. Leisure Centre
****

Next Meeting:

Wed. April 1, 1987

N.W. Leisure Centre
Starting: 7 pm