CEDAR VALLEY 99er UG NOVEMBER 1992

PRESIDENT: Jack Johns (319) 366-4541
VICE PRES: Wayne Betts (319) 377-2493
TREASURER: Bruce Winter (319) 393-0610
SECRETARY: Jim Green (319) 377-4073
NL EDITOR: Gary Bishop (319) 377-9574
LIBRARIAN: Bob Heiderstadt (319) 927-4215

CEDAR RAPIDS/MARION

Supporting the TI-99/4A and Geneve 9640 in Eastern Iowa

NEXT MEETING: 6:30 PM NOVEMBER 10, 1992

WEST MUSIC, COLLINS ROAD SQUARE

MEMBERSHIP RENEWAL NIGHT!

DOOR PRIZES!

CONTENTS PAGE

RECENTLY RECEIVED NEWSLETTERS:

Micropendium, October 1992, rcvd 10/23
West Penn, October 92, rcvd 10/30
Chicago TI UG, November 92, 10/30
Newsnetter 99, Aug Sept 92, rcvd 10/7
Chicago TI UG, Oct 92, rcvd 10/16
Cleveland Area 99 User Groups, Oct 92, rcvd 10/16
LA Topics, October 92, rcvd 10/16
Byte-Line, July, Aug/Sept 92, rcvd 10/22

MINUTES OF THE OCTOBER MEETING

There were 9 members at our meeting this month, at West Music, on October 13th. The evening got under way at about 6:50 with the treasurers' report by Bruce Winter. The minutes for our September meeting were approved, as printed, in the last news letter. Our secretary was unable to make it to our meeting this month.

We need to remember to renew our membership at the November meeting.

OLD BUSINESS: 1. Bob Heiderstadt brought in some information on a product that he had told us about, at the previous meeting. This is a liquid that you put on your contacts, such as our modules, and it will make a conductive contact that acts like solder for conductivity but the connections can be easily removed. If it works as good as their information says, this should solve our intermittent contacts on the cartridge slot. The product is sold by D.W. Electrochemicals Ltd. Canada. A 15 ml tube costs $36 plus $5 postage, one tube should be enough to coat all the TIs in our group. Gary Bishop moved that we purchase this item and bring our computers to a meeting to have them coated. We voted to make the purchase. 2. You need to let John Johnson know what type of articles you would like to have him download from the CONNI BBS, now that we are able to make downloads. John will print up a list of subjects that are available and will pass it around at one of our meetings. 3. By the time you read this the TI Fair in Chicago will be history for this year. There are at least four of our members planning on attending.

NEW BUSINESS: No new business this month.

PROGRAM: John Johnson gave a demonstration of his program that won him fourth place in the Chicago User Group programing contest. He also showed us a couple of programs: Computer Wars, which is written in 100% assembly and Robot Fighter by Mike Ward which was written in C language.

Your fill in Secretary, Bob Wahlstrom

I have played with Johnathan Guidry's NEW program on the next page, and it operates exactly as he claims. The only small quirk I can find is in the amount of memory is left after executing his program. When the console is first fired up, there is 11840 bytes of stack free, and 24488 bytes of program space free. After involking the NEW program, you have 11816 bytes of stack, and 24512 bytes of program space free. Not a big deal, by any means. Good work Johnathan. --Gary Bishop.

"NEWing From Inside a Basic Program----Can It Be Done?"

By: Jonathan D. Guidry
731-H Creighton Dr.
New Iberia, LA. 70560

The NEW command is very valuable to the programmer. However, it only executes in intermediate mode. Have you ever wondered how to make the computer clear the memory from INSIDE a program? If you tried to use the word "NEW" after a line number, you got a *COMMAND ILLEGAL IN A PROGRAM error. The keyword new, in intermediate mode, resets memory positions -31868 and -31952 to FF, FF, FF, FF in hexidecimal. Here is an example of the usage of resetting memory positions -31868 and -31952. (NOTE: If you plan to use the below example, the E/A or Minimem or Extended Basic and the Memory Expansion must be in place).

```
CALL INIT
CALL LOAD (-31868,255,255,255,255)
CALL LOAD (-31952,255,255,255,255)
```

In Extended Basic, chain the three statements together using the double colons (::). If you have a program in memory at the time these are executed, it will be erased. Try it. These statements are more useful at the end of a program. For example, if you place these statements at the end of a directory program called LOAD in Extended Basic, it will give you back the prompt after it is finished, and the memory will be erased.

These three statements can also guard that top-secret program you've been working on by erasing the memory if the correct password isn't given at the beginning of the program. It also prevents snooping. (Heck, I have 4 little brothers and sisters who snoop--ages 6 to 13). Also, do not forget they can be put anywhere in a program.

For those of you who want to ask me a question, send compliments, etc., please do not hesitate to write me directly or call me. Also, send me anything of use that you would think of being of help to me. I do not know everything, I am just an advanced hobbyist. (Heck, I am just starting to learn the TMS 9900 Assembly Language). Thank you for reading this and have a nice day.

Jonathan D. Guidry

Addendum pertaining to protected programs. If you want to protect an Extended Basic program using the address I gave you last month with the "NEW" command I gave you this month, use the following subroutine. (If it detects the program not being protected, then it clears the memory.)

```
30000 SUB PROT
30020 CALL PEEK(-31931,A)
30030 IF A<>128 THEN 30050
30040 SUBEXIT
30050                          CALL
LOAD(-31868,255,255,255,255) ::
CALL                LOAD(-31952,
255,255,255,255)
30060 SUBEND
```

PRETTY PLEASE, PINCH MY DEAR AUNT SALLY RUDELY!

by Jim Peterson

My apologies to dear old Sal. That mnemonic device is usually given as just "My Dear Aunt Sally", but I expanded it a bit. It is intended to remind you of the sequence in which your computer solves an equation, which is -

(P)arentheses
(P)owers (exponentiation)
(P)refixes (plus and minus)d
(M)ultiplication
(D)ivision
(A)ddition
(S)ubtraction
(R)elational operations

So what? Well, if one of your program lines isn't giving you the expected results, it may well be that you forgot to pinch Saly properly!

The computer goes through the line from left to right 5 times (I don't know if it really does, but that is the easiest way to explain it!) The first time through, it looks for a left hand parenthesis. If it finds one, it stops at the first right hand parenthesis. If it finds one but not the other, it CRASHES! When it finds a right parenthesis, it backs up leftward until it comes to the closest left hand parenthesis. It solves everything between those two parentheses, step by step in accordance with the following priorities, and then erases those two. Then it goes through the same routine again until it finds no more parentheses.

Need a "for instance"? OK -

```
X=((10*2)-6)+(8/4)
X=((20)-6)+(8/4)
X=(20-6)+(8/4)
X=(14)+(8/4)
X=14+(8/4)
X=14+(2)
X=14+2
X=16
```

Next it goes through the equation looking for the caret sign. That is the little ^ that tells it to multiply the preceding number by itself as many times as the following

number. Example -
4^2 means 4 times 4
6^3 means 6 times 6 times 6

Then, the prefixes. That just means that, for instance, if removing the parentheses from -(-6) has left you with --6, it becomes a +6, of course. I suppose that ABS and SGN are also worked here.

Now, multiplication and division. These are both done in one pass through because it doesn't make any difference which is done first. 10*2/4 is the same as 2/4.

Next, addition and subtraction, also in one pass because 10+4-2 is the same as 4-2+10.

Finally, the relational operations, which had best be the subject of a separate article. And finally finally the string concatenations, but let's keep old Sal out of those.

Note that everything between a pair of parentheses is worked as a separate equation, step by step in the above sequence, before the parentheses are erased.

So, why should you need to worry about all this?

Well -

```
10*4-2=38
10*(4-2)=20
10*4^3=640
(10*4)^3=64000
((10*4)^3=....SYNTAX ERROR!
```

Makes a difference, doesn't it?

The important things to remember are -

If you want to add two numbers together before you multiply or divide their sum, put them in parentheses (2+3)*4.

If you want to subtract one number from another before you multiply or divide the result, put them in parentheses (10-4)/2.

If you want to add, subtract, multiply or divide numbers before you increase them by any power, put them in parentheses (10*4+8)#3.

If you keep Sally in mind, you will have fewer bugs in your programs!

XMODEM
by Jim Swedlow
(taken from TI-BITS NUMBER 13 from ROM newsletter)

You may have heard of a transfer protocol calleed XMODEM and wondered what it is. If you use FAST-TERM or 4A TALK, you probably use it. The following should give you some idea of how it works.

When you communicate with another computer on phone lines through modems, your data must travel through the same voice phone lines that we use everyday. Some connections are better than others. Most have noticeable static.

Your brain, a computer whose power has never been equalled, can usually distinguish the 'data' (voice) from the 'noise' (static). It is almost impossible for your computer to make this judgement.

In the early days of data transfer, data was simply sent and the receiving computer had to do as good a job as it could to distinguish between data and noise. In a text, or DV80 file, this was not a major problem. If one character was bad you could easily find the problem and edit it.

With a memory image or Program file, however, one bad byte could render an entire file useless. Although editing is possible, it is very tricky. In August 1977, Ward Christensen developed an error detection method called MODEM2. It was also dubbed "Christensen" protocol or XMODEM. It is very simple. Data is sent in blocks of 128 bytes. XMODEM adds up the values of all the characters in each block and compares that number with a total that is sent by the sending computer. If they do not agree, the receiving computer sends a code to the sending computer and the block is transmitted again.

In 1982, Ward Christensen and Chuck Forsberg released an enhancement called Cyclic Redundancy Checking (CRC). CRC does sequential division on each character in the block resulting in a significant improvement in error detection.

Both protocols continue to be called XMODEM. Although others have been developed, XMODEM is used by all major systems, including Compuserve. (Source: an article in FOGLIGHT)

Appendix B

HP 110 IMPLEMENTATION OF MODEM7 PROTOCOL

The protocol used by the HP 110 is a block protocol, and is designed to transfer files in 128 byte blocks. A block number and checksum are included along with the data to help guarantee that the data is received correctly.

The block protocol operates as either 8-bit or 7-bit protocol. The line format is set to 8-bit, no parity, with one stop bit prior to file transmission and is restored to its original state after completion of transmission or user termination of transmission.

Each block of data being transferred is in the following format:

<soh><blk#><255-blk #><--128 data bytes--><cksum>

where:

<soh>	= 01H.
<blk #>	= binary number, starting at 01, incrementing by 1. 0FFH wraps to 00H.
<255-blk #>	= 1's complement of the block number, for an 8-bit machine.
<cksum>	= 8 bit sum of all the data bytes only; toss any carry.

All transmission errors are retried 10 times before the file transfer is aborted.

The following outlines what the program on the receiving end of the file transfer does:

- The receiver has a 10-second timeout while waiting to receive a block. Each time the receiver times out, it sends a <nak> to the sender program, and increments its error counter. The first time the receiver times out and sends the <nak>, it acts as a signal to the sender program to start the transfer.

- Once the receiver begins receiving a block, the receiver goes into a one-second timeout for each character and the checksum. If the receiver wishes to <nak> a block for any reason, it must wait for the line to clear.
- If the block numbers get out of synchronization in such a way that recovery is impossible, or if a problem arises with the mass storage device, the receiver program will wait for the line to clear, then send a <can>, thus aborting the transfer.
- When a block is received correctly, the receiver sends an <ack> to the sender program. When a error occurs, the receiver sends a <nak>, requesting retransmission of the block.

The following outlines what the program on the sending end of the file transfer must do:

- While waiting for the signal from the receiver (a <nak>) to begin transmission, the sender waits in a one-minute timeout. If the sender is not told to start within the one-minute timeout, it aborts the file transfer.
- Once the sender receives the signal to start, it will transmit a block of data, then wait for the receiver to transmit back either an <ack>, a <nak> or a <can>. If an <ack> is received, the sender transmits the next block. If a <nak> is received, the previous block will be re-transmitted. If a <can> is received, the file transfer is aborted.
- If the sender program encounters a mass storage problem, it will cancel the file transfer by sending a <can> to the receiver.
- The last block of data should contain an end-of-file character (^Z). After the last block has been correctly transmitted, an <eot> is sent, signaling the end of transfer. The sender waits for the receiver to send an <ack> signaling that it is done also.

Special characters:

<soh> = 01H
<eot> = 04H
<ack> = 06H
<nak> = 15H
<can> = 18H
<^Z> = 1AH

MY INTRODUCTION TO THE TI 99/4A

My first TI 99/4A came from a J. C. Penney store. When Texas Instruments decided to abandon their home computer, dropping the price to $50, I decided it was time for me to get one.

I read the basic book from cover to cover. Found out that I needed a small black and white TV set and a tape recorder. Those I had so I was in business. But what to do with this new toy?

Our daughter suggested she would like to have a program that would allow her to keep track of her store coupons on her TI 99/4A. That seemed easy enough. Thus began a 3 month learning session. Using only the basic book and a rough outline of what she wanted, I bravely set out to produce the most valuable program known to the computer world. Of course, others had already produced such a program, but I didn't know that. I had not purchased computer magazines and books at that time.

The first month was spent learning how to run the **### machine so that it wouldn't keep telling me that it "CAN'T DO THAT", or the various other messages it had in its inards.

The second month was spent putting together various segments that I had figured out for the program and getting them to work together. Finally, I had a program that did everything she had requested, but it used so much memory that she could only store about 25 coupons in the data bank. She sure could get any bit of information she wanted, tho!

The third month was spent removing features, rewriting sections with different, and shorter, sub-programs so that the MEMORY FULL message didn't show up. In the process, I reduced the program from 280 lines to about 180. Much later, as a learning tool, I substituted program segments from other programs, and further reduced the program to about 100 lines.

But the ironic part of this whole story is that our daughter did enter all the coupons she had into the program. Used it once and decided that her shoebox filing system was easier and faster than hooking up the TI 99-4A to her TV, entering the program from the tape player and then calling up the answers she wanted. To enter the coupons in the program, she had sorted and indexed them, setting up a simple file system that served her purposes. Chalk up one for a a learning experience, but not to the use of a home computer at home!

Bob Heiderstadt

B7 B6 B5 / BITS B4 B3 B2 B1	ROW	COLUMN 0		COLUMN 1		COLUMN 2		COLUMN 3		COLUMN 4		COLUMN 5		COLUMN 6		COLUMN 7	
0 0 0 0	0	NUL SPACE	0 0 0	DLE P	20 16 10	SP	40 32 20	0	60 48 30	@	100 64 40	P	120 80 50	`	140 96 60	p	160 112 70
0 0 0 1	1	SOH A	1 1 1	DC1 (XON) Q	21 17 11	!	41 33 21	1	61 49 31	A	101 65 41	Q	121 81 51	a	141 97 61	q	161 113 71
0 0 1 0	2	STX B	2 2 2	DC2 R	22 18 12	"	42 34 22	2	62 50 32	B	102 66 42	R	122 82 52	b	142 98 62	r	162 114 72
0 0 1 1	3	ETX C	3 3 3	DC3 (XOFF) S	23 19 13	#	43 35 23	3	63 51 33	C	103 67 43	S	123 83 53	c	143 99 63	s	163 115 73
0 1 0 0	4	EOT D	4 4 4	DC4 T	24 20 14	$	44 36 24	4	64 52 34	D	104 68 44	T	124 84 54	d	144 100 64	t	164 116 74
0 1 0 1	5	ENQ E	5 5 5	NAK U	25 21 15	%	45 37 25	5	65 53 35	E	105 69 45	U	125 85 55	e	145 101 65	u	165 117 75
0 1 1 0	6	ACK F	6 6 6	SYN V	26 22 16	&	46 38 26	6	66 54 36	F	106 70 46	V	126 86 56	f	146 102 66	v	166 118 76
0 1 1 1	7	BEL G	7 7 7	ETB W	27 23 17	'	47 39 27	7	67 55 37	G	107 71 47	W	127 87 57	g	147 103 67	w	167 119 77
1 0 0 0	8	BS H	10 8 8	CAN X	30 24 18	(	50 40 28	8	70 56 38	H	110 72 48	X	130 88 58	h	150 104 68	x	170 120 78
1 0 0 1	9	HT I	11 9 9	EM Y	31 25 19	)	51 41 29	9	71 57 39	I	111 73 49	Y	131 89 59	i	151 105 69	y	171 121 79
1 0 1 0	10	LF J	12 10 A	SUB Z	32 26 1A	*	52 42 2A	:	72 58 3A	J	112 74 4A	Z	132 90 5A	j	152 106 6A	z	172 122 7A
1 0 1 1	11	VT K	13 11 B	ESC [	33 27 1B	+	53 43 2B	;	73 59 3B	K	113 75 4B	[	133 91 5B	k	153 107 6B	{	173 123 7B
1 1 0 0	12	FF L	14 12 C	FS \	34 28 1C	,	54 44 2C	<	74 60 3C	L	114 76 4C	\	134 92 5C	l	154 108 6C	|	174 124 7C
1 1 0 1	13	CR M	15 13 D	GS]	35 29 1D	-	55 45 2D	=	75 61 3D	M	115 77 4D	]	135 93 5D	m	155 109 6D	}	175 125 7D
1 1 1 0	14	SO N	16 14 E	RS ~	36 30 1E	.	56 46 2E	>	76 62 3E	N	116 78 4E	^	136 94 5E	n	156 110 6E	~	176 126 7E
1 1 1 1	15	SI O	17 15 F	US ?	37 31 1F	/	57 47 2F	?	77 63 3F	O	117 79 4F	_	137 95 5F	o	157 111 6F	DEL	177 127 7F

* NOTE: CTRL/KEY = WHILE PRESSING THE CONTROL KEY, PRESS THE [KEY.

KEY

ASCII CHARACTER	ESC	33	OCTAL
CTRL/KEY	[	27	DECIMAL
		1B	HEX

MA-7246

Figure 5-1 ASCII Chart

NEXT REGULAR MEETING: Tuesday

November 10, 1992 6:30 PM

WEST MUSIC COMPANY
COLLINS RD. SQUARE, MARION
NORTH OF LINDALE MALL

HEAR ABOUT THE CHICAGO FAIRE
FIRST HAND

Cedar Valley 99'er Users Group
c/o Jim Green
377 Cambridge Dr. NE
Cedar Rapids, Iowa 52402-1446

FIRST CLASS

Send To:

_____