S I N C - L I N K $1.50 per copy

The Toronto Timex/Sinclair Users Club Newsletter Vol. 1, No. 3

P.O. Box 7274, Stn. A Toronto M5W 1X9

In this issue:

==========

From the Past President:

Well, the Club is a year old now, and we have 100 members from across Canada. We also have a new President, Greg Lloyd. Greg is one of the founding members of the Club and has been involved with the Club since day one. I'm sure Greg would appreciate any suggestions, comments, or ideas you can throw at him. I'll be hanging on for awhile as Treasurer and would like to hear from anyone interested in becoming our new Editor for Sinc-Link. I would like to thank everyone for making our first year such a big success.

Yours sincerely,

Pete Harvey

==========

MULTIPLICATION ACCURACY WITH THE ZX 81 - G.F. Chambers

As the Sinclair manual states, the ZX81 can multiply with an 8 digit accuracy.

The following program will allow you to multiply two integers of any (practical) length with absolute accuracy.

```
30 INPUT X$
40 INPUT Y$
50 DIM A(LEN X$ + LEN Y$)
60 FOR M = LEN X$ TO 1 STEP -1
70 FOR N = LEN Y$ TO 1 STEP -1
80 LET C = M + N
90 LET B = VAL X$ (M)
100 LET A(C) = VAL Y$ (N) * B + A (C)
110 LET I = INT (A(C)/10)
120 LET A (C) = A (C) - (I* 10)
130 LET A (C-1) = A(C-1) +I
140 NEXT N
150 NEXT M
160 CLS
170 PRINT X$; " X " ; Y$; " = ";
180 PRINT
190 FOR P = 1 to (LEN X$ + LEN Y$)
200 PRINT A (P)
210 NEXT P
220 STOP
```

==========

Newsletter prepared by GRAPHIC ENTERPRISES, WORD PROCESSING CENTRE

ADVANCED MACHINE CODE PROGRAMMING

by Stan Piotrowski

This article assumes you have some knowledge of Machine Code programming and as such will not explain the basics of machine code other than where necessary. There are several good books on the market explaining machine code, such as Toni Baker's book "Mastering Machine Code on Your ZX-81".

With just a little knowledge of machine code and a few machine code commands, you are actually capable of programming and this article shows some of the uses of those commands. The problem with the books on teaching machine code is that they tell you everything, what each command can do, but not a step by step detailed explanation and a working program.

Whenever working with machine code, you are well aware there are no Syntax checking for errors as in Basic, so you may get something that does not work or a crash. If a program is not doing what it is supposed to do, always remember that it works in LOGIC and logic only. That means the only alternative is to get out your pencil and paper, make headings up of addresses used, stacks and your registers then go through each command line by line, writing down what the computer would be doing, jumping to or Pushing or Poping and erasing where numbers are to be erased and see what the end result is. It is the only way I do it and everytime something will not work, I check it out on paper going through it logically and always find what went wrong.

Whenever you work in machine code where you will be putting in extended programs, work in stages and try each stage out until you have all the bugs out, then join the stages together into a working program. If you disassemble some small programs such as the Scroll routines in my last Newsletter article, you can find out what the logic is and how some of the commands are being used. This is the way I learned some of my machine codes.
I made up headings with the addresses, registers, etc. and wrote each line out as explained previously. Okay, onto some uses of your learned machine code commands.

Every machine code programmer has used Basic and entered arcade-type games with the result that they are very, very slow. By the end of this article you will be able to move a graphic character around the screen at super high speed and I do mean fast.

The 1st couple of reminders are that as you have read, the HL and the A registers are the preferred registers in the Z-80A microprocessor and these will be used quite frequently.

Since we will be dealing with TV display, let's start with the display variable. In your manual, you will by now have come across the various variables from 16383 to 16513. There are two that we will be dealing with in this article and the first one is 16396. There is a very in-depth article by Harry Doakes in the July/August issue of "Sync" so I will very briefly describe how we make use of this address. This address holds the location of the address location where the display file is located since the display file moves around in memory. In a long program, it could conceivably be located at around address 32000. This is why if your program is supposed to be 8K long, you will get addresses printed at a higher address location. We use this address to obtain the exact location so that we can print anywhere we want onto the screen. Another reminder is that we all know the screen (including the bottom 2 lines) holds 24 lines X 32 columns or 768 possible locations for any one of the 64 Sinclair characters. But things are not that simple in machine code land. In Basic, we can print at 10, 15 and know that the character will be somewhere near the centre of the screen. In machine code we have to deal with code 118 which occurs at the beginning of the display file and after each line, so now we have to remember 32 columns plus 1 (end-of-line code). It makes calculations a bit more difficult. We will be using two register pairs to display a character....HL and DE (or BC if you like).

```
42, 12, 64      LD HL, (16396)
17, 76, 0       LD DE, 76
25              ADD HL, DE
54, 128         LD (HL), 128
201             RET
```

Let's go through each line logically (there's that word again) and in detail.

continued.....

Line 1: we load HL with the contents of the location of the display file. Remember that brackets are like a box containing something so we say "the contents of" and in this case we have to use HL & we are loading HL the address location where the display file begins. Without brackets the line would read to load HL with that decimal number.

Line 2: we load DE with the decimal number 76 where 76 is a location on the screen at about line 1 column 15. I will let you work out the columns and lines and do not forget the code 118 where required.

Line 3: we add DE to the contents of address HL which all boils down to this. If the address location of the display file begins at say, 30000, then from 30001 (remember code 118) to about 30792 are all the possible locations of displaying a character on the screen. (again remember code 118 after each line.) We want the location 76 so this is added to HL.

Line 4: we now load the contents of HL or in other words, that address location with the character of the code 128 which in this case is inverse space. Don't forget the meaning of the brackets. If they were left out, you would load HL with the decimal number 128 and cancel out the number it contained after adding DE to HL.

Line 5: returns us to the original call which in this case is Basic. Code 201 can also be used within a machine code routine such as 16514 - Call 17000. The machine code immediately jumps to 17000 and begins executing the machine code instructions. It goes on until it reaches a return instruction such as 201 and immediately jumps back to 16517. (16515 and 16516 holds the address location.)

Enter the above program using your loader program or the one from my last newsletter article. I will assume you know about characters in the first REM statement otherwise again read about it in my last newsletter article.

Now, RAND USR 16514 and immediately an inverse space will appear.

That was pretty simple so let's get it to move. (I will be explaining only new lines added and you can go back each time to read the explanation of the old lines.)

```
42, 12, 64          LD HL, (16396)
17, 34, 0           LD DE, 34
25                  ADD HL, DE
54, 61              LD (HL) 61     : letter X
35                  INC HL
126                 LD A, (HL)
254, 0              CP 0
32,2                JRNZ +2
24,246              JR -10
201                 RET
```

Line 5: we add 1 to the display file location. So if the display file is as in the above program at 34 which is line 1 column 1 we now are at number 35.

Line 6 and Line 7: we want to compare something in line 7 and we can only use the A register to compare something with so we have line 6. Now the reason we want to compare is that we don't want to run off the end of the screen and cause a crash. So we load A with the contents of HL which could be any character code or the end-of-line character code. We now test it in line 7. If the character is 0 which you know is a space then continuing to line 8.....

Line 8: we jump over 2 bytes if the comparison is not 0. Remember when we do a comparison, the computer subtracts the comparison with the contents of A. If it is not zero or space such as code 118 (0 - 118 is definitely not zero). If it is a space (0 - 0 = 0) then it skips over the next command where it comes to line 9 which tells it to jump back 10 bytes to LD (HL), 128. By the way, line 9 shows 246 which to the computer means jump back 10 bytes. If you, for example, POKE 20000 with - 10, the computer will automatically convert it to the positive decimal number.

continued.....

If you enter you will immediately get 1 line full of \'s. Machine code is so fast that you will not see it check and print each \.

But this is not moving graphics. There is one thing to keep in mind and that is moving graphics is an optical illusion. We print one character then erase the last one. So let's change the above program to do this. We will also add a delay loop to slow the process down and you can change the delay loop to see how fast machine code is.

The first 4 lines above are the same so continuing with line 5:

```
229             PUSH HL
1,1,10          LD BC 2561
11              DEC BC
120             LD A,B
177             OR C
32,251          JR -5
35              INC HL
126             LD A, (HL)
254,0           CP 0
32,8            JRNZ +8
54,61           LD (HL),61
225             POP HL
54,0            LD (HL),0
35              INC HL
24,233          JR -23  :to PUSH HL
225             POP HL
201             RET
```


Line 5: we have to store the old address so one way of doing this is to push HL onto the stack. We could also store this number in HL at an arbitary address. For example, when I program, I leave the first 20 addresses (16514 to 16534) for this purpose. So instead of Pushing HL I would LD (16514), HL. Then whenever I need this number I just call it back or in the above program, we will POP it off the stack.

Line 6 to 10: this is a machine code delay loop which I used from Toni Baker's "Breakout" game. (You see the value in disassembling machine code routines.) To increase speed simply decrease 10 to a lower number and to slow down, change 10 to a high number.

Line 15: note that we again print another X before erasing the old one. In machine code this occurs so fast that we have the illusion of movement. If we erased the old one before the new one was printed, then we get a flashing effect which is disturbing to the eyes.

Line 16: here is where we POP the old address of the stack which contains the location of the first X or previous X that was printed. We could have simply decreased HL (DEC HL) but in larger programs that old address could have changed by as much as a couple of hundred locations so get used to storing numbers for this reason.

Line 17: now that we have the old address we can erase "X" so we load the contents of the display in HL with 0 or space. So at this point in time an X is now printed at the next location, 36, while the old location, 35 has been erased.

Line 18: we now increase the HL location again so that when in the next line we jump to push HL, the new address will now become the old address.

Line 20: after the X has reached code 118, it JRNZ to this line. What we are doing here is clearing the stack. If you follow the program through you will find that HL has been pushed onto the stack and we cannot go all day long pushing without having an even number of poping since the stack does have a limit of the number it can hold and the program would eventually crash. So this line clears it leaving it empty.

continued.....

Now you should be capable of moving the X to the left side of the screen. (By changing INC HL to DEC HL). By combining both programs your X will move to the right then the left and on and on. This leaves us with one more step & that is keyboard control of graphics.

Now for the next variable address we will be using is 16421. This with 16422 holds the codes of the key being presently pressed. I will explain it a bit and give you a BASIC program to obtain the codes a little later on.

What we will do now is add a few more lines to the previous program and give both right and left movements. The following is the whole program. There is a slight difference in that instead of PUSHing and POPing the display file location, we will store those numbers at specific addresses - 16514 to 16517. So this time, write to address 16518 and remember when you are finished to RAND USR 16518.

```
42,12,64            LD HL (16396)
17,76,0             LD DE,76
25                  ADD HL, DE
34,130,64           LD (16514), HL
34,132,64           LD (16516), HL
```

(The last two lines simply store the value of HL which will be used in the program to locate the new location and erase the X at the old location.)

```
54,61               LD (HL),61
205,187,2           CALL 699              :keyboard scan routine
235                 EX,DE,HL
33,127,253          LD HL,                with the codes for break or space
167                 AND A                 :clear the carry flag
237,82              SBC HL,DE
200                 RET Z                 :return to basic if you press break
1,1,5               LD BC,1281            :Change 5 to 0 for fast
11                  DEC B
120                 LD A,B
177                 OR C
32,251              JRNZ -5
205,187,2           CALL 699
235                 EX DE,HL
33,247,223          LD HL,                code for key "5" or left arrow key
167                 AND A
237,82              SBC HL,DE
32,25               JRNZ +25              :to next K-Scan
```

(if it is not obvious by now, you can see that 699 address is a ROM routine that fetches the codes of the key you are pressing and returns the value in HL where we exchange it into DE).

We then load HL with the value of the key we wish to tell the computer to act on and in the first case it was the SPACE key while above it was the "5" key. If HL matches DE (result is zero after subtracting) then the computer will act. If it does not match it moves on to the next K-Scan (JRNZ +25). In the case of the SPACE key, it returns us to Basic otherwise you stay forever in machine code until the computer is shut down.)

Continuing on with the program.....

```
42,132,64           LD HL,(16516)         :fetch the display location
```

continued.....

```
43                  DEC HL              :decreasing HL allows us to move left
126                 LD A,(HL)
254,0               CP 0                :is it a space?
32,16               JRNZ +16            :jump to next K-Scan if no space
34,132,64           LD (16516),HL       :store new display location
54,61               LD (HL),61          :print an X at the new place
42,130,64           LD HL,(16514)       :get the old location
54,0                LD (HL),0           :erase the old X
42,132,64           LD HL,(16516)       :get new location
34,130,64           LD (16514),HL       :store it as an old location
```

Move X to the right:

```
205,187,2           CALL 699
235                 EX DE,HL
33,239,247          LD HL,              Codes for key "8" or right arrow
167                 AND A
237,82              SBC HL,DE
32,25               JRNZ +25            :to the end of program
42,132,64           LD HL,(16516)
35                  INC HL
126                 LD A,(HL)
254,0               CP 0
32,16               JRNZ  +16           :to end
34,132,64           LD (16516),HL
54,61               LD (HL),61
42,130,64           LD HL,(16514)
54,0                LD (HL),0
42,132,64           LD HL,(16516)
34,130,64           LD (16514),HL
195,149,64          JP 16533            :jump to first K-Scan
```

Ensure the numbers are correct if it does not work. This program has been tried by me so in printing the newsletter there may be errors. If you checked & it still does not work, check the mnemonics in your manual for the correct codes.

Before we get into the keyboard scan routines, did you notice how we used addresses to store the display location of HL instead of PUSHing them? The reason behind this is for you to get used to storing addresses and when you experiment in moving vertically you will appreciate this since instead of just INC or DEC HL, you must LD DE,33(remember code 118) and ADD HL,DE. This location would then be immediately above or below the old location. (Don't forget AND A to clear the carry flag before SBC DE,DHL when moving up.)

If you wish to obtain the codes of the keys in case you would rather have some other character on the keyboard to be used for movement such as "A" left and "L" right use the following program to obtain them: (# means space)

```
10  IF INKEY$=""THEN GOTO 10
20  LET K$=INKEY$
30  LET A = PEEK 16421
40  LET B = PEEK 16422
50  SCROLL
60  PRINT "CODE FOR#";K$;"#is#";A:",";B
70  GOTO 10
```

Copy the codes down so you can refer to them when you need them.

With a couple more commands of the over 500 commands available in machine code, you should be able to write a complete arcade-type game. I did so I'm sure you can.

==========

A MODEM FOR THE TIME (X-1000 or ZX-81)

by Franz Hrazdira

After looking at many advertisements in various magazines, I found an offer which I was willing to explore further. Having been reassured on the phone that the modem kit would require nothing more to function satisfactorily than about (4) hours of my time to put it together, I ordered it right then. This was back in April.

continued....

When I received the package from Byte-Back, I was impressed - not so much with what I had paid after the exchange rate, duty and tax was added. as with the quality of the parts and documentation. Two separate manuals, covering the MD-1 modem and the RS-232 board in great detail, were supplied.

The BAUD RATE, which is the speed at which the RS-232 board sends and gets data, determined by (1) the frequency of the transmit and receive clock and (2) the divide rate of the INTEL 8251-A USART IC chip (Universal Synchronous/Asynchronous Receiver Transmitter), can be set to either 9600, 4800, 2400, 1200 or any of these baud rates divided by 16 or 64 - the most common being 300, which is 4800/16. The schematic makes available, in addition to the RS-232 level signals, also the TTL level signals for interface to any printer or other peripheral.

The MD-1 uses the new Motorola MC 6860 modem chip which MODulates and DEmodulates your signal so you can communicate with any other computer over the telephone. To do this, you first load and run a terminal program which comes on tape as part of the deal and takes up about 1.5K of your memory. Then, assuming everything was assembled correctly and the modem is connected to the phone, you can just dial up any of the following numbers on this list.

423-3265	ETI	between	5PM	and	9AM
366-2069	CFTR		6PM		9AM
978-6893	MEDICAL		7PM		9AM
499-7023	IBMPC		24	Hours	
868-4000	DATAPAC				
365-9621	COMPUSERVE				
424-1895	for further details or a demo				

Normally, you will be connected after a few ring tones. The other computer starts out with a 2000 Hz answer tone from its modem to which yours sends a 1000 Hz originate tone in reply. You are now ready to communicate by entering NEWLINE, and away you go.....

A new terminal program which is now available lets you send and receive all kinds of data, including program in BASIC and Machine Code, while doing an automatic syntactic sum check for error free transfers. After receiving, you can then review, print or save the data on tape for later use.

This program also has an automatic print mode which, if selected, will send the other computer a command to wait. After everything was printed, it will then ask for more data to be sent, to receive a new screen full and so on, until you change the mode.

Let me tell you also about a service you can subscribe to by phone, which will give your little computer the power you have only dreamt about until now. It is called Compuserve. For $5.00 US an hour, you can send and receive Electronic Mail from among 70,000 other subscribers in Canada and the US, chat on a multi-channel CB simulator with someone in Alaska, B.C., or California, post notices on a number of Special Interest Group (SIG) bulletin boards, play multi-user games, read about commodity futures or search a database on just about any topic.

I am offering starter kits to Compuserve for $60.00 CAN. They include a very basic manual, a three ring binder, a user I.D., a password, a subscription to the TODAY magazine, regular update bulletins and (5) five hours of free connect time to the service. The modem will cost you $198.00 CAN in kit form or $246.00 assembled and tested, including the basic software.

For more information, you can call me at the last number on the list.

==========

TRICKS OF THE TRADE

By J.J. Castillos

After a few years of working with the ZX81, it is only natural that we learned ways to protect our programs from being appropriated or altered by other people.

The tips that follow will not stop a determined and knowledgeable person from replacing your name or copyright notice with his or her own or prevent everyone from peeking and using your machine code routines, without proper acknowledgement, but they will make it very difficult for the average ZX81 owner to do these things.

Suppose you put in your program a line such as the following: 1 REM PACMAN XVII - JEAN SMITH, 1983. It is very easy to delete it and replace it with another, but if you enter as a command POKE 16510,0/Newline, you will see that your line 1 has become line 0 which cannot be deleted or edited out.

Now, if your aspiring pirate knows this trick, he will just have to POKE 16510,1/Newline and the line will be again at his mercy for quick disposal.

However, there is a way to make it very hard, if not impossible, for him to do this. Just begin your program with your REM statement as line 1 and then POKE 16509,62/Newline. Your line 1 will become line F873 and if you start to add other program lines, you will see that your REM goes to the bottom of the listing, safe from 99% of other users, undeletable and uneditable. In order to get to your REM line, people would have to either delete the whole program to get it back to the top of the screen and reverse the process or disassemble the program to locate the address of the REM line number.

On the other hand, if you have machine code that you want to protect from predatory eyes, you can put it into a REM statement at the beginning of your program and then make it invisible by entering POKE 16514,118/Newline. Users trying to list the program will only see the line number but nothing else. You can even do better by entering instead of the above, POKE 16509,120/Newline. This will make the whole line invisible as if it didn't exist, users will naturally assume that the program begins elsewhere (i.e., the next BASIC line you enter which will be visible).

An alternative which is just as effective, is to put your machine code as an array which you dimension, define and after the computer has accepted it, then you delete the BASIC lines (DIM, LET, etc.) that defined it. Thus, the array with your machine code is in memory to be transferred wherever you want it to go but to the uninitiated it just doesn't exist. If he tries to run the program, it will be wiped out.

But let us now reverse the situation and suppose that with all good intentions it is you who want to alter or see what there is in a program as part of your learning process or to make improvements. You can see the invisible machine code in the first line by identifying (use PRINT PEEK command), which of the first few bytes of the line (16514,16515, etc), contain a number 118 and then POKE there any number such as 0 or 20. Try to list the program and if you eliminated all the 118's you will see it all displayed on the screen. If address 16509 is not 0, make it 0, using a POKE command.

If the programmer used some of the more elaborate tricks such as arrays, you can PEEK blocks of memory of say, 200 bytes each, until you find the machine code you are looking for. For the purpose, run a simple program such as the following:

```
 5 FAST
10 REM ENTER STARTING ADDRESS FOR THE SEARCH
20 INPUT A
25 PRINT A;":  ";
30 FOR B=0 TO 200
40 PRINT PEEK (A+B);"-";
50 NEXT B
60 STOP
```

continued...

Start if you wish your search at location 20000, after the first run try 20200, etc. For most of the memory you will probably only get strings of 0's, but wherever you see numbers instead, it could be the machine code you are looking for. The above is, in fact, a short and simple disassembler program.

Finally, if you wish to make a back-up copy of one of those "unlistable" and "unsaveable" machine code programs in the market, you can easily do it by not using the normal LOAD command. Set your computer first in FAST mode, enter then RAND USR 837/Newline and start your tape recorder in play. You will see that the program loads but instead of starting to run and lock itself in machine code, it stops with an error code. Make your back-up copy using SAVE & run by entering GOTO (whichever line number that has a USR Command)/ Newline.

So Far, we don't know of any ZX81 program that has been successfully protected from being copied or peeked in the ways described above, which is no doubt a constant worry to software manufacturers but a blessing to us users, who most of the time just want to learn from more knowledgeable people, or be protected from accidentally erasing the only copy of a valuable program.

==========

HARDWARE AND SOFTWARE REVIEWS

By J.J. Castillos

There are so many new add-ons and programs for the TIMEX 1000/ZX81 that we could not possibly review them all in detail in this Newsletter. With this in mind, I thought it would be more useful to Club members, to run from time to time brief reviews of many items we have bought and tested, so that people can have at least some guidelines to help them pick the right product from among the bewildering array of ads in the computer magazines.

HUNTER BOARDS These memory boards provide up to 8K of RAM or EPROM (2K or 4K) memory in the 8-16K area which is not normally used in the ZX81. The RAM memory boards include battery back-up, which allows you to keep programs in them for up to 20 years per battery even when the computer is shut down. These boards are sold as kits which are easy to assemble by anyone with a minimum degree of electronic expertise. The instructions are very well written and printed and have many useful tips to use and even improve your boards. A fine product indeed sold at a very reasonable price of US$ 29.95.

MEMOTECH'S MEMOCALC - This spreadsheet is supplied as an EPROM in another of those elegantly designed Memotech add-ons and with a 64K RAM pack, it allows you to have as many as 7000 numbers simultaneously on the grid. It has a switch to connect you to BASIC or Memocalc directly. You see at any one time 3 columns and 17 lines but your screen display works as a window which can be moved up or down, left or right at will, to anywhere you want. You can save, print out modify or re-start grids at any time. This is an attractive product selling for only US$ 49.95 but frankly, if you have the latest, machine code version of VU-CALC, you don't really gain so much as to justify the extra expense.

MCODER - This program is a compiler, ie., it translates your BASIC programs into machine code programs which run up to 20 times faster. This compiler is somewhat limited in the sense that it allows you only one single dimension numerical array, it works with integers only, no multiple conditional statements are allowed etc., but it only takes 2K of memory, it is not infuriatingly frustrating as other compilers (ZXPRESS for example) and if you are resourceful enough to adapt your programs to meet its requirements, it compiles very easily (you only have to press a key!) and points to any error you may have made so that you can correct it and go on. The instructions are concise but adequate. MCoder has received somewhat negative reviews in most magazines but my experience with it has been positive and I find it very useful.

EVOLUTION - We have all probably heard of those British Columbia kids who wrote a game based on the theory of evolution for an APPLE computer and made a bundle with it. This English program for the ZX81 lacks the graphics, colour, sound and speed capabilities of the APPLE version but it is reasonably fast and it is also an excellent didactic experience to familiarize people with the main facts of evolution, at the same time that it offers a real challenge to your intelligence. It is one of my favourite games and I recommend it to anyone who wants to go beyond the dozens of zap'em & blow-them-up games currently available.

==========