

# QB MONITOR

QB-99'ERS U.C. NEWSLETTER

4

March 1988

The QB MONITOR is the Newsletter of the QB-99'ers User Group, is printed Sept. thru June and sent in exchange for other User Group Newsletters. Send Exchange Newsletter to Frank Cotty, Queensborough Community College, Bayside, NY 11364. Credit original sources.

The QB 99'ers meets the second Saturday of each month September through May, at Queensborough Community College, Bayside New York, room S225. See the calendar at right for the dates. TIME 2 PM.

March 1988							April 1988						
S	M	T	W	T	F	S	S	M	T	W	T	F	S
			1	2	3	4						1	2
6	7	8	9	10	11	12	3	4	5	6	7	8	9
13	14	15	16	17	18	19	10	11	12	13	14	15	16
20	21	22	23	24	25	26	17	18	19	20	21	22	23
27	28	29	30	31			24	25	26	27	28	29	30

NOTE DATE

Are you receiving our newsletter regularly? If you check the top line of the address label on the envelope containing this newsletter the months listed are those we have received your newsletter. To continue to benefit from exchange with our group your group must send a newsletter within three months beginning March 1, 1988.

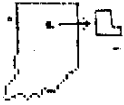
## HAPPY ST. PATRICKS DAY

Contents	Page
Editor's note .....	1
Assembly Made Easy .....	2
Editor's Note:	

As you have already noticed from the contents listed above I have opted for one article. The article is on assembly language. This article will probably explain for you all the things that many authors assumed you knew. This article was recommended by Ed Machonis, so you know it must be well written, intelligent and easy to follow. If you follow the directions in this article I guarantee you will improve your assembly language knowledge. Even if you never program a line in assembly again you will benefit from this article. "Assembly Made Easy" Courtesy of the Miami Co. Area 99-4A II C U G of Indiana, August 1987 issue.

Articles for the April issue must be in by March 15

In the February issue of the LA TOPICS newsletter we noted the news of George Steffen, an active TI'er, returning to New York for surgery and cancer therapy. The QB 99'ers is extending the courtesy of complementary newsletters as our "Get Well" message. I am asking all TI 99'er User Groups to do the same. Send your newsletters to Mr. George Steffen 5935 Siebig Avenue, Bronx, NY 10471. Get Well Soon George!



# MIAMI CO. AREA 99-4A H C U G



The First Recognized 99-4A User Group in Indiana

P O Box 1194

Peru, IN 46978

Vol. 5 No. 8 August 1987

## \* ASSEMBLY MADE EASY \*

I am sure that you have noticed that I have been including some Assembly Language routines in the past few issues of this newsletter. Before I go any further, I must point out that I am not an A/L programmer! The routines that I have included have either been copied from another source or were a close imitation of someone else's work. My purpose in including these A/L routines was to introduce you to the power and speed of this very powerful language. After asking those at our monthly meetings if anyone had experimented with these programs, I discovered that in most cases, no body had given them a try.

A possible cause for this reluctance to jump on the A/L bandwagon is a lack of understanding of how to enter and assemble the code. I will use the rest of this issue to try to rectify this situation. If you will take the time to follow along with me through these examples, I think that we can take some of the mystery out of entering and assembling A/L code. To quote the TI-Writer manual: "Let's do it!".

## \* FIRST, THE TOOLS \*

Before we get started there are a few basic requirements. Your equipment will have to include: at least one disk drive and 32K memory expansion. The TI Editor Assembler package will not be required because I will assume that you all have a copy of The Funnelweb Farm Utility Loader. I am using version 3.4 of this program and it is available from our software library.

Finally, you will have to have some means of loading the Funnelweb software. For most of you this will be the Extended BASIC cartridge. Other possibilities include the CorComp disk controller, Mini Memory, or the Editor Assembler cartridge.

## \* THE CONCEPT \*

Unlike programming in BASIC where you can enter your code and then type "RUN" to see your program execute, A/L programs first require you to enter the code with an Editor and then you must convert that code into machine language with an Assembler. The code that you enter with the Editor is for you to read, while the code that the Assembler generates is for the computer to read. Furthermore, BASIC is full of helpful error messages to help you to recode your program when a problem is present. The only error messages you get with A/L are those that are detected by the Assembler when it is assembling your code. The E/A package includes a Debugger program to help the experienced programmer to analyze any program errors not detected by the Assembler, but the debugger's use is beyond the scope of this article



## QB MONITOR ~ QB-99'er NEWSLETTER

### \* LET'S ENTER THE PROGRAM \*

Now it's time to dust off the keyboard and enter the program listing found later in this newsletter. First turn on your system (it works better when powered up) and then load The Funnelweb software. When the first menu is displayed select EDIT/ASSM and then EDITOR from the next menu. Your drive will run and a TI-Writer type text editor will load. With the cursor on the command line, select T for tabs. Notice where the tabs are set (don't change anything), They are preset to allow you to use the Tab key (FCTN/7) to jump to the proper field when entering your code. At this time you should also notice the hollow cursor indicating that word wrap mode has been disabled. This keeps your computer from reformatting your code into one large paragraph. Another difference from the TI-Writer Editor is the character set used. This Editor does not use the true lower case characters like TI-W does so you can tell at a glance which Editor you are using. Finally, when you use Save File to save your program, the extra baggage that TI-Writer writes to disk is not included. To my knowledge everything else is the same.

To get out of the command line, press ENTER. We are now ready to start typing our program. Notice that the line number is displayed next to your cursor. I prefer to turn the line numbers off with FCTN/zero, this is a personal preference and is up to you (the same keypress will turn the numbers back on again). Make sure the Alpha Lock is down and start entering the program exactly as listed. Be sure to use the Tab Key (FCTN/7) to advance to the next field and remember that Comments are optional and need not be typed.

Don't forget to save your program every few lines. This will require that you have a fresh data disk in the proper drive. To do this, enter the command line by pressing FCTN/9, then type SF for save file. Type in the correct drive number and use the file name MORSE/S then press enter. Another thing that should be done from the command line (if you have a printer) is to list your file to the printer so you can check for errors. To do this type PF for Print File and then type in your printer name (PIO or RS232..... etc.)

When you are sure the code is entered correctly, save it to disk (DSKn.MORSE/S). Now exit the Editor by pressing FCTN/9 to get to the command line, then type Q, then ENTER, then E, then ENTER.

### \* NOW WE'LL ASSEMBLE YOUR CODE

Making sure the Funnelweb disk is in the drive from which it was originally loaded (it remembers), select ASSEMBLER from the menu. The drive will spin and the Assembler will load into memory. Once loaded, you will notice another nice feature of Funnelweb, your file name (MORSE/S) will be displayed as the default Source file and the default for the Object file will be MORSE (Funnelweb truncates the last two letters of the source file to give you a suggested name for your Object file). If these first two selections are correct press ENTER twice until you get to LIST DEVICE NAME. If you are planning to list your code (I usually don't) enter either your printer name or a disk filename here and press ENTER. For those of you who use PIO for your printer name, you will have to add a period (PIO.) to make things work correctly. If you do not wish to list your code, just press ENTER to advance to the OPTIONS selection.

## QB MONITOR ~ QB-99'er NEWSLETTER

You will see two options listed (R and C) The R option is almost always used and should be selected here, it tells the Assembler that your Source Code was entered using R to designate register numbers. The C option will save your Object Code in a compressed format, saving disk space and decreasing loading time. If, however, your program will be loaded as a CALL LOAD, CALL LINK from Extended BASIC, you can't use the C option because XB will not load a compressed file. Other possible options are L (list), S (symbol table), and the undocumented T (text). L tells the assembler that you want a listing directed to the device named above, S creates a symbol table following your listing, and T prints out the location and hex code for all the text strings in your code. For this example we will select the default R and C. Now start the assembly process by pressing PROC'D (FCTN/6).

If everything went correctly, after the Assembler is finished, you will receive a message saying that there were no errors. If errors were detected, the type of errors and the line number on which they occurred will be listed. To correct errors you must reenter the Editor, load the Source Code, correct it, resave it, return to the Assembler, and reassemble the code again. Depending on your typing and proofreading skills, this could take several times. Now you see why people buy Ram Disks.

### \* IT'S TIME TO SEE IT RUN \*

When you get your code assembled without errors it is time to run your program so you can admire your work. To do this first exit the Assembler by pressing ENTER. This will bring you to another menu where you can select LOADERS. This again will bring up another menu where you will want to pick the LOAD/RUN (E/A) option. Once loaded from the Funnelweb disk, this loader will prompt you for a file name to load. Our file will be the object file that you just assembled (DSKn.MORSE). If, however, you forgot your file name, Funnelweb comes to the rescue again. Any time you are not in one of the Editors, Funnelweb will allow you to catalog your disk by pressing FCTN/7. You will be looking for a Display/Fixed 80 file to have the loader run for you.

Now back to business, you are being prompted for a file name, type in DSKn.MORSE. Again you will be prompted for a file name (you can have more than one file in memory at once), just press ENTER. Now another nice feature of Funnelweb is on the screen. The E/A cartridge would prompt you for a Program name here and you would have to know the name of the entry point where you wanted the program to start running from (START in this case). Funnelweb goes one step further and reads the REF/DEF Table and displays the program entry points for you at the top of the screen. To start your program running, just use the arrow keys to place the cursor under the desired entry point (some programs have several) and press FCTN/6 (PROC'D).

At this point, your program will be running. Press any of the letter keys (with the Alpha Lock down) and you will hear the morse code for that keypress. When you want to exit the program, simply press ENTER and it will return you to the title screen. Congratulations! you have entered, assembled and ran your first Assembly Language Program. You are now an Expert!

\* LET'S MAKE IT AUTO RUN \*

With minor changes we can change the original listing to make the program automatically run once loaded into memory. To do this load the original code into the EDITOR and change line 0101 from:

0101           END

to:

0101           END   START

Once this is done, save it to disk as DSKn.MORSE1/S, then enter the Assembler and assemble it like before using the R and C options.

Once assembled, this program should run as soon as it loads in. This bypasses a few keypresses and eliminates the requirement for someone who is using the E/A cartridge to know that the program entry point is START. For programs with multiple entry points this would not be good practice but our program example is a natural for this technique. I included it here to show you that some A/L programs use a different procedure to execute.

\* LET'S MAKE IT RUN FROM X-BASIC \*

In my opinion Assembly Language is most useful when it is used to write routines that can be called from Extended BASIC. This will allow you to write most of your code in a language that most of us feel comfortable with and reserve A/L code for those chores that require the increased speed or the special features that A/L can offer. On the negative side, this procedure requires that you first load your XB program and then have it load your A/L routines. This will add to the time required for your program to be up and running. Now, however, even that drawback is history, as there are now methods available that allow you to save your XB and A/L codes together in one file. The Load Program in Funnelweb is an example of this. In fact, Funnelweb includes a utility (LEG/S) on the disk to do this for you.

Let's alter our original Source Code (MORSE/S) so we can load it from XB. First we must understand three facts of life about XB's Assembly Language Loader. First it does not know how to handle Ref's such as those found on line 0005. These routines must be equated to an address. A list of these Equates can be found starting on page 415 of the E/A manual. Secondly, when you are running an A/L routine from XB, you must provide a method of adding >60 to any character that you wish to display on the screen. The chart found in last month's newsletter has a column where this conversion was done for you. The program (MORSE/S) that we are working with has no text or graphics displays so we are not concerned about this. Finally, when we assemble a routine that will be called from XB, we can't use the C (compressed) option because the XB loader will not load compressed files.

Now let's do it! Once again load MORSE/S into the Editor. Remove line 0005 (REF KSCAN,SOUND) with FCTN/3 (DEL LINE) and then insert two lines after the original line 8 (DOTIME EQU 4500) with the FCTN/8 key. On these two inserted lines add Equates for KSCAN and SOUND as shown below.

## QB MONITOR ~ QB-99'er NEWSLETTER

### \* EQUATED VALUES

DOTIME EQU 4500 DELAY FOR DOT TONE  
KSCAN EQU >201C  
SOUND EQU >8400

Save this altered file as DSKn.MORSE2/S and then assemble using only the R option. Since this file was not saved as a compressed file, you might want to use FCTN/7 to catalog your data disk to see the difference in size between a compressed file and a non compressed file.

The following XB program will call the above Object Code so it can run from the XB cartridge:

```
100 CALL CLEAR
110 DISPLAY AT (10,1):"loading A/L code"
120 CALL INIT :: CALL LOAD("DSKn.MORSE2")
130 DISPLAY AT(10,1):"depress ALPHA LOCK":"Press any Letter Key"
140 CALL LINK("START")
```

The above XB program uses CALL INIT to initialize memory, then loads the assembly code with CALL LOAD, and starts it running with the CALL LINK statement. When preparing this article I tried to get the A/L code to auto run once loaded like we did in the second exercise but I was unable to do this. After a couple of phone calls I found out from Jerry Rowe that XB will not allow this unless you trick it by AORging your A/L code into the ISR hook at >83C4 to get it started and then clearing out >83C4 early in the program so it will not keep reloading itself every 1/60th of a second. I tried this and it worked but it seemed like alot of work just to avoid using the CALL LINK statement.

### \* ONE FINAL EXERCISE -

Up until this point we have been using the Assembler to create DF-80 Object code files in either compressed or uncompressed format. With a few alterations to the original code and with the use of Funnelweb's FWSAVE utility, we can create Program Image files. Program Image files are the most compact form of A/L code that the 99/4A supports. They are also the fastest loading files but they require a special loader. When using Funnelweb, go to the Loaders menu, and you will find options 1 and 3 (TEXT MODE or PROGRAM) both of which will load Program Image files. Another loader came to us last week as Fairware from Paragon Computing. This loader called EASLOAD, will load Program Image files from XB.

As stated above, Program Image files are the most compact and the fastest loading of all TI files. They also will automatically start running once loaded. If a Program Image file is longer than 33 sectors it will be split up into two or more files. Every file after the first will have a name that ends with a character one ASCII code higher than the last character in the previous file (ED-->EF-->EF etc.). The utility that creates Program files from DF-80 files handles all this for you, I just mentioned it so you would understand what you are seeing on some disk catalogs.

## QB MONITOR ~ QB-99'er NEWSLETTER

Again, let's do it! Load MORSE/S into the Editor and make the first 10 lines look like those below and then change the original line 101 to look like the last line listed below. Save this altered file to disk as DSKn.MORSE3/S and then load the Assembler and assemble it using the R and C options.

```
      IDT 'MORSE'      TRANSLATE CHARACTERS TO MORSE CODE
      DEF SFIRST,SLAST
*
* EXTERNAL REFERENCES
      REF KSCAN,SOUND
*
* EQUATED VALUES
DOTIME EQU 4500      DELAY FOR DOT TONE
*
SFIRST LWPI WS      INITIALIZE WORKSPACE

SLAST END
```

Nest back out of the Assembler and go to the Loader menu. Select the Load/Run option and load DSKn.MORSE3, THEN PRESS enter, next type DSKn.FWSAVE and press ENTER twice. Select SAVE as your entry point and follow the prompts. When done your file will have been converted to a Program Image file and stored under the name that you selected. Once again you will want to catalog your data disk and see the size of this last file compared to the previous files.

Next go to the Loader section of Funnelweb and load this last file with option 3. Be sure to notice how fast it is loaded and that it will start running as soon as it is loaded.

### \* SOME CLOSING THOUGHTS -

This series of exercises ended up taking more space than I had anticipated. In fact, this is one of the reasons that this newsletter is late getting to you. I feel that it would be worthwhile for all of you to take the time to run through these examples so that you will have some idea of how things work in Assembly. If you have any trouble, let me know and I'll try to get you back on track. See you at the next meeting!



QB MONITOR ~ QB-99'er NEWSLETTER

```

0001      IDT  'MORSE'      TRANSLATE CHARACTERS TO MORSE CODE
0002      DEF  START
0003      *
0004      *  EXTERNAL REFERENCES
0005      REF  KSCAN,SOUND
0006      *
0007      *  EQUATED VALUES
0008      DOTIME EQU 4500      DELAY FOR DOT TONE
0009      *
0010      START  LWPI WS      INITIALIZE WORKSPACE
0011      GETKEY SB  @>8374,@>8374      SELECT ENTIRE KEYBOARD
0012      BLWP @KSCAN      CHECK KEYBOARD
0013      MOVB @>837C,R0      READ KEYBOARD STATUS
0014      COC @KEYMSK,R0      CHECK KEYBOARD STATUS
0015      JNE GETKEY      JUMP IF NO KEY YET
0016      MOVB @>8375,R0      KEY PRESSED, PUT ASCII CODE IN R0
0017      ANDI R0,>7F00      STRIP OFF PARITY BIT
0018      CB R0,@CHARA      COMPARE CODE TO "A"
0019      JL NALPHA      JUMP IF NOT ALPHABETIC, MAY BE CR
0020      CB R0,@CHARZ      COMPARE TO "Z"
0021      JH NOGOOD      JUMP IF NOT ALPHABETIC
0022      MOV R0,R3      COPY CHAR CODE TO R3 (LEFT BYTE)
0023      SWPB R3      PUT CHAR IN RIGHT BYTE
0024      AI R3,-65      SUBTRACT CODE FOR "A" = INDEX
0025      SLA R3,1      MULTIPLY INDEX BY 2
0026      MOV @MCTABL(R3),R4      GET TABLE ENTRY IN R4
0027      MOV R4,R3      COPY TABLE ENTRY TO R3
0028      SRL R3,8      RIGHT JUSTIFY ELEMENT COUNT
0029
0030      SENDEL LI R10,>9100      TURN ON
0031      MOVB R10,@SOUND      TONE
0032      CLR R2      PUT ZERO IN R2
0033      SRL R4,1      SHIFT NEXT ELEMENT CODE INTO CARRY
0034      JNC DOT      JUMP IF DOT
0035      AI R2,DOTIME*2      ADD DELAY FOR DASH
0036
0037      DOT AI R2,DOTIME      ADD DELAY FOR DOT
0038      BL @DELAY      DELAY AND FNDTONE
0039      LI R2,DOTIME      GET INTER-ELEMENT DELAY TIME
0040      BL @DELAY      DELAY AFTER ELEMENT
0041      DEC R3      DEINCREMENT ELEMENT COUNT
0042      JNE SENDEL      JUMP IF MORE ELEMENTS TO SEND
0043      JMP GETKEY      ELSE GO GET ANOTHER CHARACTER
0044
0045      NALPHA CB R0,@CHARCR      IS CHARACTER A CARRAGE RETURN?
0046      JEQ EXIT      IF SO, GO EXIT
0047      NOGOOD LI R10,>F400      TURN ON
0048      MOVB R10,@SOUND      NOISE
0049      LI R2,DOTIME+2      SET DELAY TIME FOR NOISE
0050      BL @DELAY      DELAY AND TURN OFF NOISE
0051      JMP GETKEY      GO GET NEXT CHARACTER
0052      EXIT BLWP @0      GO HOME
0053
0054

```

QB MONITOR ~ QB-99'er NEWSLETTER

```

0055 DELAY SRC R12,15      KILL TIME
0056      DEC R2          DECREMENT DELAY TIME
0057      JNE DELAY      JUMP IF MORE DELAY
0058      LI R10,>9FFF    TURN OFF
0059      MOV B R10,@SOUND TONE
0060      SWPB R10        TURN OFF
0061      MOV B R10,@SOUND NOISE
0062      B *R11         RETURN TO CALLER
0063 *
0064 * DATA CONSTANTS
0065 *
0066 KEYMSK DATA >2000    KEYMASK
0067 CHARA TEXT 'A'       CHAR CODE FOR "A"
0068 CHARZ TEXT 'Z'       CHAR CODE FOR "Z"
0069 CHARCR BYTE >0D      CHAR CODE FOR CARRIAGE RETURN
0070 *
0071 * TRANSALTION LOOK-UP TABLE
0072 *
0073 MCTARI DATA >0202    A= .-
0074      DATA >0401      B= -...
0075      DATA >0405      C= -.-.
0076      DATA >0301      D= -..
0077      DATA >0100      E= .
0078      DATA >0404      F= ...-
0079      DATA >0303      G= --.
0080      DATA >0400      H= ....
0081      DATA >0200      I= ..
0082      DATA >040E      J= .---
0083      DATA >0305      K= -.-
0084      DATA >0402      L= .-..
0085      DATA >0203      M= --
0086      DATA >0201      N= -.
0087      DATA >0307      O= ---
0088      DATA >0406      P= .---.
0089      DATA >040B      Q= ---.-
0090      DATA >0302      R= .-.
0091      DATA >0300      S= ...
0092      DATA >0101      T= -
0093      DATA >0304      U= ...-
0094      DATA >0408      V= ....-
0095      DATA >0306      W= .--
0096      DATA >0409      X= -.-.-
0097      DATA >040D      Y= -.-.-
0098      DATA >0403      Z= -.-..
0099 *
0100 WS      BSS 32      WORKSPACE
0101      END

```