TBX Handbook v1.7
© Spencer 2015

History

In 1968 Honeywell decided to produce a 12 bit computer the H-112 which was meant for industrial installations and so was actually called the H-112 Controller rather than computer, though in fact it was very similar to the DDP-516 and H-316, 16 bit computers manufactured by Honeywell with a slightly reduced instruction set and only 12 bits per word rather than 16.

The H-112 wasn't very successful and in the early 1970's I obtained one of these computers which was being scrapped.

The H-112 had a maximum memory of 8KW and a basic teletype interface which I modified to work with a 5 unit Baudot code Creed 7ERP telex machine. Later I upgraded my home system with a Digital Group 32 x 12 line Video Display and Ascii Keyboard and my own designed interface. At the beginning Programs and Data were saved onto 5 unit Paper Tape and later onto cassettes.

Like most computers you could program it in an Assembly language and it had a program called CAP12 which was the H-112 assembler or a program SAP12 which ran on a series 16 machine to assembler for the H-112.

But there were no high level languages like Fortran, Basic or Cobol. In 1974 the first micro computers were appearing but expensive and with very small memories so a magazine called Dr. Dobbs Journal began pushing the idea of a 'Tiny Basic' that is an 'Integer' only Basic for these small machines and I thought why not make a somewhat Extended version for my H-112 and the result was TBX 'Tiny Basic Extended' a high level language interpreter for the H-112. Which was announced in an issue of Dr Dobbs Journal in 1974.

For any questions or comments I can be contacted via my E-Mail address
h112@spencerweb.net
And photos of the machine can be seen at :-
www.spencerweb.net/Ian_and_Julie/Hobbies/Chips_-_Early_years/chips_-_early_years.html

TBX Structure

The main TBX Editor and Interpreter resides in the bottom 4K of the computer and the TBX program and Array's reside in the top 4K. After loading the TBX interpreter is started by doing a 'Master Clear' then setting 'RUN' and pressing the 'START' button. After initialising the following message appears :-

Tiny Basic Extended Revision BH
.

The '**.**' Indicates that the TBX editor is ready to receive input.

TBX Statements

TBX Statements always have the form :-

Line number Instruction Parameters
Ex**.** **10 LT A=6**

Multiple statements/Line are permitted :-
Ex**.** **10 LT A=6 : GT 300**
Separated by a '**:**'

Pressing any key while a program is running will cause a **BREAK** unless Break has been turned off.

TBX Variables

The language allows single letter variables A to Z but with or without an index 0 to 9. So in effect there are 260 variables. So all of the following are valid variables.

A L A5 B3 etc.

Each variable can hold a value between –2048 and +2047 (12 bits)

Random Number

! provides a random number between 0 and 999, Example :-

10 LT A=!/100

produces a random value for ‘A’ between 0 and 9

Array’s

The area of memory above the TBX program can be used as an Array either for numeric and/or text data and is represented by the designator ‘**@**’ for numeric values and ‘**&**’ for text characters.

Numeric Array

So for example @0 or @220 could contain a value which could be used just like the normal variables A to Z.

10 LT @10=230

Text Array

Or you can use parts of this area for ‘Text’ for example with &0 set to a word like ‘TEST’, actually when used for text each array variable contains 2 characters so in &0 there is ‘TE’ in &1 there is ‘ST’ and in &2 an ‘End of text marker’ which is a carriage return (6 bits/character using Min6b format). So you can allocate any number of array locations to contain your text.

10 LT &0=”TEST” (Set &0,&1,&2 to TEST plus C/R)
10 IN &0 (Input Text to &0,&1 etc.)

If for example your messages could each be up to 15 characters then you could allocate &0 to &7 for the first message &8 to &15 for the second and so on. Printing &0 would print out the complete first message, printing &8 would print out the complete second message etc.

Mathematics

TBX handles 4 basic mathematical operators + - * /
And always executes them left to right, so :-

10 LT A=6+4*10

is 100 not for example 46

Commands

When a '**.**' is shown on the screen you can enter Commands or TBX instructions. The commands are as follows :-

NEW
Clear the Program and Data areas

CLEAR
Clear only the Data area

MAP
Display the length of the TBX program and the space available for Array's

LIST x,y
List the TBX program onto the screen or **LIST x,y** lists the lines between Line x and Line y

RUN x
Run the TBX program from the first line or **RUN x** run the program from Line x. **RUN** clears all of the Data variables

CONTINUE x
Continue without a line number continues running the program from the break point with a Line number it's like RUN without clearing the Data variables

LOAD
Load a TBX program from the cassette mounted in the cassette drive

SAVE
Save a TBX program to the cassette mounted in the cassette drive

Instructions

LT
This is the let statement so all of the following are valid statements :-
10 LT A=6
10 LT @20=200
10 LT &50=”TEST”
10 LT A5=B*C2+D
19 LT SQ=26 (The contents of Q are an index for S so S0 to S9)

IN x
This is the Input statement all of the following are valid statements :-
10 IN A
10 IN B9
10 IN @30
10 IN &50

PR
This is the PRINT statement a print statement may be terminated in a number of ways :-

10 PR “TEST” (Causes a New Line after the PRINT finishes)
10 PR “TEST”. (Leaves the cursor on the next character position)
10 PR “TEST”; (Leaves one space after the PRINT finishes)
10 PR “TEST”, (Tabs to the next tab after the PRINT finishes)

All of the following PRINT statements are valid :-
10 PR
10 PR “TEST”
10 PR z9
10 PR @30
10 PR &20

When in **DG Display mode** (see the **DM** Command) then the following are also supported :-

10 PR $C. (Clears the screen, Cursor placed at 0,0)
10 PR $5,10. (Places the cursor on 6th line the 11th character)

GT x

This is the GOTO statement all of the following are valid :-

10 GT 100
10 GT A7
10 GT @25
10 GT A+B

GS x

This is the GOSUB statement it has the same parameters as the GoTo and the subroutine must be terminated with a Return statement

RT

Return from a Subroutine

IF x : y

This is the IF compare statement and if the statement is true then all of the following statements on the line will be executed.

Operators are

= Equals
Not equal
>Greater than
< Less than

So all of the following are valid statements :-

10 IF A = B : GT 400
10 IF C7 # D : GS 700
10 IF SQ < @50 : GT 300
10 IF @15 > A5 : LT A=B*6 : GT 600

Random numbers and Mathematics are not allowed in an IF statement.

FR w=x,y,z

This is the FOR statement and has the parameters starting value then end value and step value. The FOR loop must be terminated with a Next statement

All of the following are valid :-

10 FR A=1,5,1
10 FR A=0,50,10
10 FR A3=B,C9,D6
10 FR A7=@20,60,D7

NX w

The Next statement at the end of a FOR/Next loop.

ON x,y,z

The ON statement if x=1 then Line y is executed if x=2 then Line z etc.

10 ON A,100,200,300,500

SL x
This is the sleep statement and is number of $1/10^{th}$ of a second to sleep, so for example :-
10 SL 20 (Sleeps for 2 seconds but the exact time depends on the speed of your PC)

ST
The program should always terminate with a STOP statement, failure to do this results in an ERROR 3 Message.

RM
A remarks line which is ignored by the program, if it contains space characters then it must be enclosed in quotes.
10 RM "THIS IS A REMARK"
10 RM THIS-IS-A-REMARK

BN x
Turns on the BREAK function.
A break is caused by entering any key on the keyboard while the program is running.
if **x = 0** then interrupt the program and return to the command processor when a break occurs, this is the standard condition when a program starts running.
If **x<>0** then it is the line number jumped to when a break occurs.
10 BN 2000

BF
Turns off the BREAK function so that the program cannot be interrupted

BK
Force a jump to the break routine

RB
This is RETURN FROM BREAK and should be the last statement of a Break routine.
Execution continues from the next statement after the break.

AS x
Execute an Assembler routine starting at location x (decimal). This address is always in the top 4K and should be above any TBX program or Array data. The assembler code must be in the form of a subroutine with the first location empty for the return address and the routine must end with a jmp* (indirect) through this location.

TM
Put the Emulation in Teletype mode. After a program terminates the emulator automatically returns to Teletype mode.

DM
Put the Emulation into DG Video Display mode (20 lines x 64 characters)
After a program terminates the emulator automatically returns to Teletype mode.

***** The following commands are not implemented in this version of TBX for the Emulator *****.

CH Chain program as only one program/memory area is stored on each cassette.

TS Test sense as this was used to test sense lines on my homebrew interface

RD/WR Read or Write data to/from cassette (These commands may be implemented later)

ERROR MESSAGES

.0001 Invalid Command
.0002 Invalid Instruction
.0003 Invalid Line Number
.0004 Invalid Command Parameter
.0005 Missing Character Position
.0006 Invalid $ Command
.0007 Invalid Variable Identifier
.0008 Non-Numeric I/P to a Numeric Variable
.0009 Variable Index Out of Range
.0010 Invalid Mathematical Operator
.0011 Missing " Terminator
.0012 Invalid IF Operator
.0013 Divide by Zero
.0014 Subroutine Stack Overflow
.0015 Return Before Gosub
.0016 Invalid Separator
.0017 FR/NX Stack Overflow
.0018 NX Before FR
.0019 Write Failure
.0020 Write Protect Error
.0021 Read Error
.0022 Non-Existent Cassette or Invalid content Error