REPORT FROM JAPAN

Taylor Jackson

Fusjisawa, 251 Japan

In Japan, basic has been around for several years now, and among hobbists Tiny basic in one version of another is probably the most widely used. Unfortunately there hasn't really been a good 6800 version around, that is until recently. ASCII magazine (Japanese) has recently published two great Tiny interpreters.

In July '78 "GAME" was released. This is an expanded version of VTL with some interesting bells and whistles (do, until-for next loops, arrays) it takes up about 1.5K for the interpreter alone. With a monitor, save-load (load to any address) and string editor it goes up to about 2.5K. The main features of this interpreter are it's speed and stinginess with memory. (about 20-30%, less memory required than full keyword type interpreters.)

Next in April '79 NAKAMOZU Tiny Basic (NTB) was released. This is the first Tiny interpreter written for a CRT based system to come out.
The main features common to both GAME and NTB are:

1. Do until loops
2. For next loops
3. 1 dimension arrays
4. Peak and poke commands
5. Calls to link with machine language subroutines
6. Decimal and Hex. can be used freely
7. Real time input
8. They are FAST

Using the bench mark tests from kilobaud I came up with the following results:

Test	1	2	3	4	5	6	7
Game	1.3	3.5	7.5	8.5	10.5	18	23
NTB	.8	7.5	14	13	17	26	37
Pittman Tiny	a	37	61	62	83	280 b	c

a: no for next loops
b: used counting loop to replace for-next
c: no arrays

I ran these on a Hitachi 6800 @1MHZ

As you can see the days of making a cup of coffee between input and response are gone.

Game is the obvious winner in the speed race but this is due to using "system variables" rather than keywords. However this test doesn't really give a true indication of NTB's speed. In actual applications (longer programs) NTB should perform much better.
For example in the case of GO TO, or GOSUB, to a higher line number NTB will begin the search from the present line rather than from the beginning.

Also searches through the statement and function tables are very time consuming. In the case of A=1+B*C+10*D, this would usually require 1 check of the STATEMENT table and 5 checks through the Function Table. In this type of operation each table has to be checked to the end.

NTB will caryy out these searches only when necessary so there should be a noticible gain in speed on longer programs. (I found it about 2.5 times faster than the PALO ALTO Tiny on a friends 8080)

The Do until and For Next loops also will allow a much cleaner program, giving a further increase in speed.

Now let's look at the general characteristics.

	Interpreter size	number range	type
GAME	1.5 K	-32768/+32767	integer
NTB	3 K	" / "	"
Pittman Tiny	2.5 K	" / "	"

Not so much difference in size. They are all well in the Tiny category. But if we look at the instruction sets for them the differences will show up.

Pittman Tiny

Commands	Statements	Functions	Variables
*Clear	*End	Run	A - Z
*List	*GoTo	USR	
*Run	*GOSUB		
	*IF THEN		
	*INPUT		
	*LET		
	*PRINT		
	*REM		
	*RETURN		

single statement per line

NTB

Commands	Statements	Graphic Statements
*AUTO	DATE	*COPY V-RAM topprinter
*APPEND	DO	*CLR
*DEL	UNTIL	*CURS
*EXIT	END	*NEG(Reverse back-ground+display)
*LIST	FOR-TO	*!W(x,y) turn bit xy on
*LOAD	NEXT	*!B(x,y) " " " off
*NEW	STEP	*!R(x,y) reverse bit xy
*RUN	GOSUB	
*SAVE	GOTO	
	IF	
	LET	Graphic Functions
	INPUT	!P(x,y) read bit x,y

```
POKE                    off=0
PRINT                   on =1
REM                     char=100
RESTORE
RET
STOP
THEN
```

```
Functions   16Bit Functions   Formating
 ABS            AND           USING
 GET$           OR            TAB
 KEY            XOR           CHR$
 MOD                          HDF
 # (PEEK)                     (4 digit hex)
 READ                         HDT
 RND                          (2 digit hex)
 SGN
 USER

 multiple statements  O.K.
```

GAME

```
Commands                  Statements
 0 or /n list(from n)     A=B Let(not written)
 #=1 run                  #=100 Goto 100
 &=0 new                  !=100 GoSub 100
 &:0)=$FF open file*      ]     ret
 &:0)=$F0 lock file*      >=n   user(N)
 == search end of file    ;=(A=1) IF
 =n change program        J=2,20 For J=2 to 20
    start* address        Next J-Step 2  @=J+2
 *=n change ram end       Do     @
 #n, old string, new      Until A=2  @=(A=2)
 string,  edit line

Functions
 'n      Rnd(n)
 %(A/B)  MOD(A,B)
 +n      ABS(n)
 #       Not
```

GAME

```
Input          Output
 A=? input A    ?(n)=A  output least n digits
 A=$ input              (leading 0 supressed)
     character          of A
                ??=A    output A (4 digits Hex)
                ?$=A    output least signifi-
                        gant byte of A
                        (nex-2 digits)
                $=A     output ASCII character
                        for least signifigant
                        bit of A
                .=A     output n spaces
                /       CR LF
              "STRING"  output quoted string
```

*program pointers

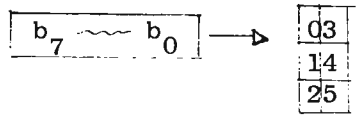
```
| EOF MARK |
 (=)                    Cold start
 (&)                    or after new command

 | PROGRAM | EOF |      After input
 (=)        &
```

When EOF=$FF program can be written into
When EOF $FF program cannot be written into

*Multiple programs may be loaded throughout available memory or run from programs in ROM When changing to a new program the starting address is set by inputting =start address and the EOF is found and set by ==.
The new program is then ready to go this way multiple programs may be placed throughout the memory. (no linkage available though)

You can see from this that GAME has some interesting features and NTB supports all of the PALO ALTO TINY instructions (abbreviations are the same also) plus having many additional features.

The graphic commands in NTB are quite useful but also graphics and special symbols may be mixed freely with
This is done by shifing the keyboard into a "graphic"mode thru software.
At present we are using a cyclic 4 stage shift;-ASCII JIS (Japanese characters), Graphic, Control Code. (no control code on our pocket keyboard). This, of course, can be easily rewritten to meet your own needs.
In the graphic mode the lower 4 bits are output directly to the screen as



This allows interesting graphics to be mixed with text.
The video ram we are using at present is very similar to the TRS-80 video board, and allows this type of limited graphics.
The graphic commands will have to be adapted to your own video board but this should be easy to accomplish.

I/O Routines

Both GAME and NTB are using MIKBUG type I/O. The I/O parameters are handled in ACCA. The other registers should be preserved. They can be run on a teletype but NTB should really be run on a video system capable of graphics, to utilize it fully.

Fast service and Feedback, almost unheard of in this modern world.

GAME

Save Load Routine

This routine will allow you to load GAME programs into any open area in the available memory.

Assembler

A 2 pass assembler written in GAME

Diassembler

To list GAME programs into a more readable form e.g. #=100 becomes GO TO 100

String Editor

Rather than retype a whole line the editor will replace all occurances of the old string with the new one in the line.

Also GAME 3.6 is out now. This is a 4 K. Graphic Version of GAME. It has the same instruction set as GAME. and features 34 graphic commands. However the graphic section was written for The Hitachi 6800 micro and video board. So, a lot of rewriting would be necessary to get it up and running.

In the near future ASCII plans to release a GAME compiler and a NTB compiler. (GAME 8080 is out in compiler form now so the 6800 version should follow soon.) Also a screen editor for NTB has been promised. I'll let you know about them as they are released.