

GRAPHIC PROGRAMMING

HP Saturn Assembly Language

By Marcos Navarro

PREVIEW

Copyrighted Material

For HP49G, HP49G+ and HP50G Calculators

GRAPHIC PROGRAMMING

HP Saturn Assembly Language

By Marcos Navarro

For HP49G, HP49G+ and HP50G Calculators

CONTENT

1. PRELIMINARY NOTIONS	1
1.1 The HP Saturn microprocessor	3
1.2 When to use Assembly Language?	3
1.3 Why study this language?	3
1.4 Resources needed	4
1.5 Editing Graphic Objects on a PC	5
1.6 Graphic Objects or Grob	5
1.7 Internal representation of a Grob	7
1.8 Grobs and bytes	14
E1 Compiling and decompiling Graphic Objects	20
H1 The Yorke chip	22
 2. SCREEN GROBS	 23
2.1 Screen Areas.	25
2.2 Screen Grobs.	25
2.3 The Main Area Grobs Registers.	26
2.4 Secondary Area Grobs Registers.	27
2.5 Pointer Concept.	27
2.6 Main Bitmap Registers.	27
2.7 Secondary Bitmap Registers.	28
2.8 Special Addresses..	29
2.9 The D0->Row1 Subroutin.	30
2.10 Loading a Grobs on the Screen.	31
2.11 Reading data from a Grob on the Stack.	33
E2 Using the Debugger to Explore Registers.	36
Experiment 2A.	36
Experiment 2B.	38
Experiment 2C.	41
H2 The Display Control System.	43
 3. SCREEN REFRESH	 45
3.1 The Right Margin.	47
3.2 Screen Refresh (Part I).	49
3.3 Scrolling the Window (Part I).	53
3.4 Screen Refresh (Part II).	55
3.5 Scrolling the Window (Part II).	56
3.6 Screen Refresh (Summary).	57
3.7 The Screen and the Stack Grob.	59

3.8 The PICT Grob	60
E3 Right Margin and Left Margin Registers.	61
Experiment 3A.	63
Experiment 3B.	63
Experiment 3C.	63
Experiment 3D.	64
Experiment 3E.	65
Experiment 3F.	66
H3 Display Connectors.	67

4. THE ANNUNCIATORS **69**

4.1 The Annunciators.	71
4.2 Annunciators Control Registers.	71
4.3 Programming the Annunciators.	72
H3 LCD Screen.	77
Parts of a LCD.	77
Light Polarizers.	78

5. SCROLLING THE WINDOW **79**

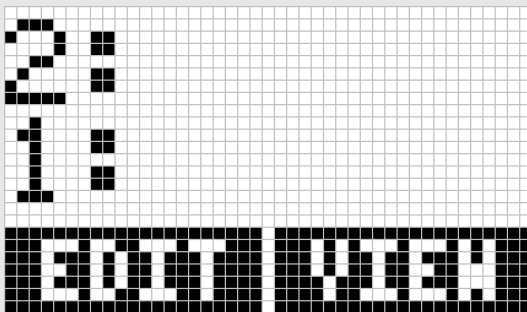
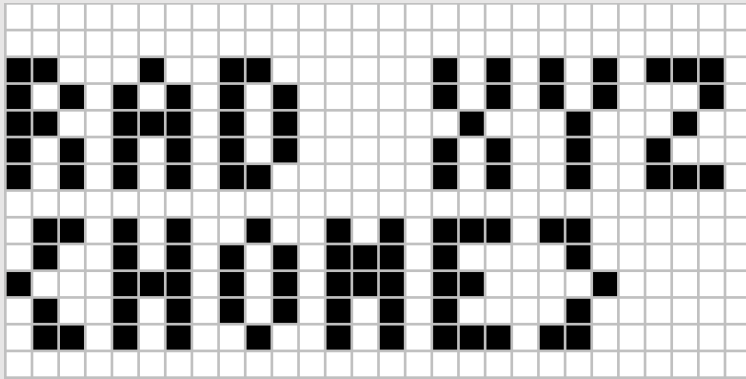
5.1 Scrolling the Window (Part II).	81
5.2 Scrolling the Window (Part III).	86
5.3 How does it work?	87
5.4 When the Left Margin is greater than 3.	89
E5 Smart Window Scrolling.	93
H5 Display Panels	95
Front Panel or columns Panel.	95
Rear Panel or Rows Panel.	95

6. THE SPRITES **97**

6.1 What is a Sprite?	99
6.2 Sprite Drawing Modes.	99
6.3 Sprite Data Reading Techniques.	100
6.4 Reading data from the Stack.	100
6.5 Embedding Sprites's Bitmaps Directly in Code.	101
6.6 Sprites on Stack Grob.	103
6.7 Columns of Nibble.	104
6.8 Using other Sprite insertion modes.	108
6.9 Copying Screen Areas.	109
6.10 Coodenates and Screen Areas.	115
6.11 Drawing Sprites on specific Screen Area.	120
H6 Perimeter seal and Panels interconnection.	125
Conduction lines between panels.	125
The Perimeter Seal.	125

7. WORKING AT THE BITS LEVEL	127
7.1 Working at the bit level.	129
7.2 Logical Operations with Registers.	129
7.3 Unset specific pixels.	129
7.4 Set specific pixels.	131
7.5 Bits-Level Shifting.	132
7.6 Moving a Sprite at the Pixel Level.	133
7.7 Inserting a Sprite into the Screen with x, y coordinates given in pixels.	142
7.8 New Coordinate System.	142
7.9 Encoding and decoding of coordinates.	144
E7 Exploring Memory with the T2 Tool.	151
Experiment 7A.	151
Experiment 7B.	152
Experiment 7C.	152
 8. VIRTUAL SCREENS	 153
8.1 Virtual Screen.	155
8.2 Drawing Sprites on a Virtual Screen.	158
 9. THE MENU	 163
9.1 The Menu	165
9.2 Drawing a custom grob in the Menu Area.	165
9.3 Virtual Screen for the Menu.	166
9.4 Drawing a custom label in the Menu Area.	170
E9 The Lines Counter.	172
 10. DISPLAY CONTRAST	 177
10.1 Display Contrast.	179
10.2 Using the fifth bit of contrast.	180
10.3 Adjust the contrast by a specified amount.	182
10.4 Loading a specific contrast value.	183
 11. GRAYSCALE GRAPHICS	 185
11.1 Grayscale Graphics.	187
11.2 How do Grayscale Graphics Work?	187
11.3 Specific memory areas for Grayscale Graphics.	188
11.4 Memory Space distribution in the HP49G.	188
11.5 Memory Space distribution in the HP50G.	189
11.6 Using the Interrupt Service Routine for Grayscale Graphics.	191
11.7 General Procedure.	192

APPENDIX	197
EXPLORING THE INSIDE OF THE SYSTEM.	199
I. The First Journey.	199
II. Where is the RTI?	200

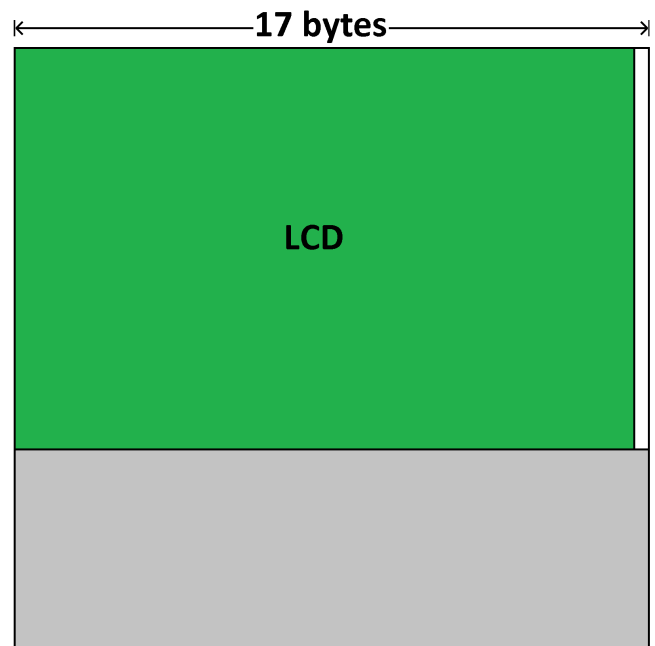
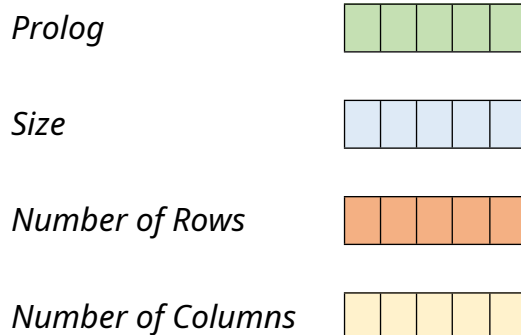


The images displayed on your graphic calculator's *Screen* are composed of a grid of tiny squares. Each of these tiny squares is a pixel. A pixel is the smallest individual point in a digital image. The word "pixel" is a contraction of "picture element".

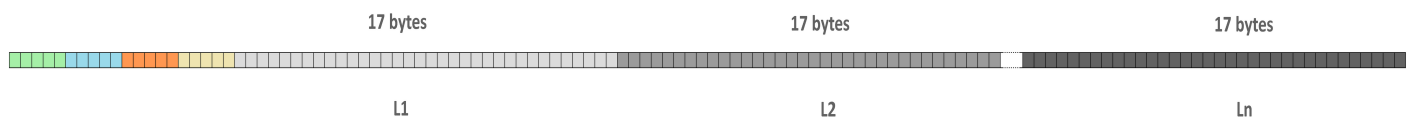
3.1 The Right Margin.

Let's review the *Internal Encoding* of a *Graphic Object*. Suppose we have a *Graphic Object* of 131 x n pixels.

The first 20 nibbles are the *Header* of the *Graphic Object*:



After the *Header*, the *Bitmap* data begins. Each horizontal line of the *Screen* is encoded using 17 bytes. The following scheme illustrates the *Internal Encoding* of this *Graphic Object*:



The first 17 bytes after the *Header* correspond to the first horizontal line of pixels at the top of the *Screen*, and the last 17 bytes correspond to the last horizontal line at the bottom of the *Screen Area*.

Although 131 pixels are less than 17 bytes, *The System* always encodes the grob in multiples of 8 bits (bytes). Unused bits are padded with zeros. The remaining bits appear as a white stripe in the image above, at the right of the *Liquid Crystal Display* (LCD).

Sometimes, we have a *Graphic Object* whose *Rows* are much longer than the *Screen*. The segment of the *Row* that is after the first 17 bytes is named the *Right Margin (RM)*. The following scheme illustrates this case:

E3

Experimentation

Right Margin and Left Margin Registers.
(HP49G, HP49G+ and HP50G)

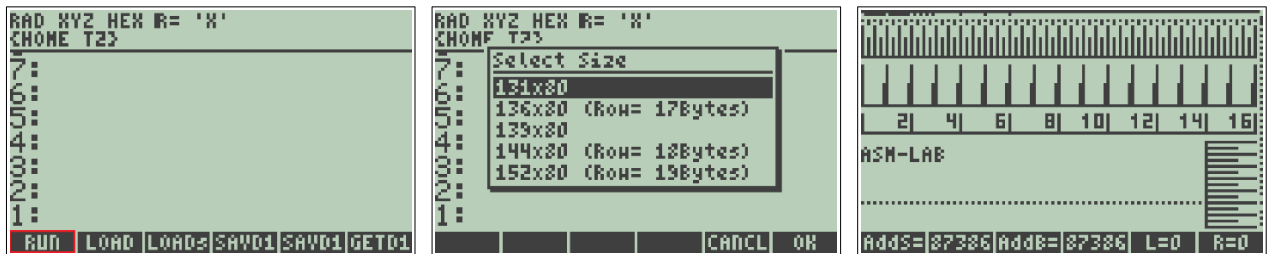
Download the *Experimentation Tools* package from the following link:

<https://sites.google.com/view/pgs saturn50g/home>

Inside the downloaded file you will find several programs designed to help you better understand the topics covered in this book.

For the experiments in this section, follow these steps:

- 1) Transfer the file **T2.HP** to your calculator.
- 2) On the calculator, run the RUN file, which is inside the T2 directory.
- 3) Select the dimensions for the *Background Graphic*, and press OK.



- 4) You can now modify the *Margins*, while seeing the effects on the *Screen*. This tool can also be used to explore the calculator memory and see the data represented as pixels.

KEY	FUNCTION
X	9 Rows Jump Up.
.	9 Rows Jump Down.
G	32 Rows Jump Up.
J	Jump 32 Rows Down.
H	Jump 64 Rows Up.
K	Jump 64 Rows Down.
Yo	128 Rows Jump Up.
L	Jump 128 Rows down.
UP	1 Row Jump Up.
DOWN	1 Row Jump Down.
LEFT	Jump 1 byte up.
RIGHT	Jump 1 byte down.

LC 00014	% 14h= 20d.
A=A+C A	% Skip the <i>Header</i> .
	% A=@StackGrob +20d
 C=R4 A	 % C= x = Horizontal jump (HJ).
A=A+C A	% A= @StackGrob + 20d + x
 C=R3 A	 % C= 34(y-1) = Vertical Jump (VJ).
A=A+C A	% A= @StackGrob + 20d + x + 34(y-1).
 D0=A	 % D0 point to nibble in <i>Row</i> y-1, <i>Column of Nibble</i> x
	% of the <i>Stack Grob</i> .
C=0 W	
A=0 W	
 *LoadBMap	 % Read data from the <i>Stack Grob</i> , <i>Row</i> by <i>Row</i> , and
	% write it to a new grob.
 LA FFE01	 % FFE01h= 1111 1111 1110 0000 0001.
C=DAT0 A	
A=A&C A	
 C=0 A	
C=DAT1 B	% B= 2 nibbles = 8 pixels
 C=C+C A	
C=C!A A	
 DAT0=C A	
D0+34	% Next <i>Row</i> of <i>Stack Grob</i>
D1+2	% Next <i>Row</i> of new grob.
	% The number of nibbles in a <i>Row</i> is always an even integer.
B=B-1 A	% Decrement the Counter.
GONC LoadBMap	
 LOAD	
D1+15	
D+3 A	
 RPL	

FIND THIS BOOK ON:

[Amazon.com](https://www.amazon.com)

[Google Books](https://books.google.com)

[Google Play Books](https://play.google.com/store/books)