manual HRG-ms 2.7

Features

This software HRG-ms is a graphic tool for the Timex-Sinclair TS1000 or Sinclair ZX81. It gives you a graphic resulution of 256x192 pixels on the screen an on the printer. The only hardware requirement is a graphic capable RAM-pack.

The software consits of following modules:

- graphics driver
- tools-library
- interface to Basic
- interface to machine-code
- hotkeys for screen-selection
- hotkey for print-screen
- hotkey for UDG test

All Basic commands are simple and easy to understand.

Details

Program location
The software installs itself over RAMTOP and therefore needs not to be included in each user program with an ugly REM line. RAMTOP will be adjusted automaticly.

Direct control-keys
Even when a program is running the user can toggle between graphic and text-window with hot-keys. The running program will not be interrupted. When in program entry mode the user can switch temporarily to view the graphic screen. This might be helpful for debugging a graphic-software.

Print screen
At any time the user can print out the screen. Even if a program is running that does not print at all. Each graphic bank and the text window can be sent to the printer without interrupting the program.

Program inputs
You can enter an input value or even write program lines while the graphic is active.

Error messages
When an error occurs the view will be switched back to textmode automaticly to let you see the message.

Machinecode interface
A special interface is prepared to use all the functions with your own machinecode program. For further informations see the examples in this document.

Installation

Before installation make sure that RAMTOP is set to 32768. Simply make a reset or just power-on the computer. After loading of HRG-16k.p or HRG-64k.p the program starts automaticly and installs itself. It will show the following startscreen.

RAMTOP will be adjusted and will be printed on the following text screen after you press BREAK.

You might then load another program from tape without using NEW or use NEW to start entering a new program.

In case of a wrong RAMTOP value you get an error message. The program will not be installed and you will not see the startscreen.

Deinstallation

Simply use a reset or just power on your computer to remove the program.

Memory

There are two versions which differ only in memory demands. If you have a 16k ram-pack then you have to use the 16k version. In this case you can not use multiple graphic banks.

If you have more than 16k then you can use both versions. You can even use the 16k version with bankswitching.

	16k version	64k version
filename	HRG-16k.p	HRG-64k.p
jump-in for Basic (same in both versions)	32736	
RAMTOP sets to	22960	30160
left memory for Basic	6576 bytes	13776 bytes
usable graphic-banks default*	3* (24k-32k) 4 (32k-40k) 5 (40k-48k) 6 (48k-56k) 7 (56k-64k)	4* (32k-40k) 5 (40k-48k) 6 (48k-56k) 7 (56k-64k)
can be used with ram-packs	16k 32k 48k 64k	32k 48k 64k

As you see the 16k version might be more flexible because it supports up to 5 graphic banks. But it cost you more memory in the Basic region.

Caution!

The 16k version can be switched to different graphic banks even if you only have a 16k ram-pack. You can view the content of the selected memory window which will be mapped to your ram-pack because of the uncomplete memory-decoding.

Viewing is safe but any writing function (CLEAR, PIX,TEXT, etc.) will crash your system immediately.

Basic-interface

Each function call is behind a PRINT command. It has the following syntax:

```
PRINT USR 32736,FUNCTION
```

or:

```
PRINT USR 32736,FUNCTION,X,Y,...
```

You find a list with all function and parameters in the chapter **function table**.

Its a good practice to use a variable to hold the jump-in address. In the following we use a variable called HRG.

Example:

```
10 LET HRG=32736
20 PRINT USR HRG,ON
30 PRINT USR HRG,PIX,255,191
```

The name of the functions (here ON and PIX) need not to be a predefined variable. The program does not refer the function name as a number.

In this example the function ON and PIX are called. The parameters of PIX are direct behind the function name. They are all separated with commas.

The parameters are refered as numbers or as numeric variables or as any expression which evaluates a number. The range will be checked on each function call. In case of any mismatch you will get an error and the program will be stopped.

Function table

Please find more details to each function in chapter **Function description**.

name	**parameter**	
PIX,	X,Y	pixel at x,y
UNPIX,	X,Y	set, reset, invert
XPIX,	X,Y	
LINE	X,Y,U,V	linie from x,y to u,v
UNLINE	X,Y,U,V	set, reset, invert
XLINE	X,Y,U,V	
LINETO	X,Y	linie to x,y
UNLINETO	X,Y	set, reset, invert
XLINETO	X,Y	
BOX	X,Y,U,V	rectangle with edges at x,y and
UNBOX	X,Y,U,V	u,v
XBOX	X,Y,U,V	set, reset, invert
CIRCLE	X,Y,R	circle with center at x,y
UNCIRCLE	X,Y,R	and radius r
XCIRCLE	X,Y,R	set, reset, invert
POLY	X,Y,X,Y,...	draws a closed
UNPOLY	X,Y,X,Y,...	polygon with multiple edges at
XPOLY	X,Y,X,Y,...	x,y
ON		switch to graphic screen
OFF		wsitch to text screen
INVERT		invertiert graphic srceen
TEXT		text screen to garphic screen
UNTEXT		copy, reset, invert
XTEXT		or copy with background
OVERTEXT		
UDGNEW		resets the character data
UDGSET	C,B0,B1,..,B7	define a character
HCOPY		print the graphic screen
CLR		delete the graphic screen
BANK	N	select a graphic bank

Control-keys

There are four functions that can be controlled with the keyboard.

Toggle between text and graphic

To switch between text and graphic screens press keys 9 und 0 (GRAPHICS and RUBOUT) at the same time. Do not press the SHIFT key! Each time you do this the view changes from text mode to graphic mode and vice versa.

You can do so even when you are viewing the program listing. So you can compare the graphical result of a program with the corresponding lines in the listing.

Or you can create a program that outputs graphics on the graphic screen and a corresponding text in the text screen at the same time. The user has the choice to view any of both alternatively and on his own control.

The running program will not be interrupted.

Select a graphic bank

If you have more than 16k memory then your program can switch between different graphic banks to work with.

The user is able to view a different graphic bank as the program works on in another bank.

To sitch the banks just press 9 und 0 (GRAPHICS and RUBOUT) together and additionally one of the following numbers 3, 4, 5, 6 oder 7. The number represents the graphic bank.

Hint: in the 64k version key 3 has no function.

The running program will not be interrupted.

Print screen

To print the screen just press 9 und 0 (GRAPHICS and RUBOUT) together and additionally key C. The current visible screen (text or any graphic bank) will be sent to the printer.

You can trigger this even if a program runs and is not inteded to print anything.

As long as the printout is not interrupted with the break key the running program will go on after the print.

UDG test

To check some UDGs (User Defined Graphics) you can generate a quick graphic view of the actual text-screen. You do this by just pressing 9 und 0 (GRAPHICS and RUBOUT) together and additionally key T.

You have to keep T pressed as long as you want to view the UDGs in the graphic screen. When you release the key your view will switch back to the text-mode.

This hotkey is very handy if you are just entering some UDGs and want to check the result.

A running program will not be interrupted.

Basic example

The following program draws some lines on the screen.

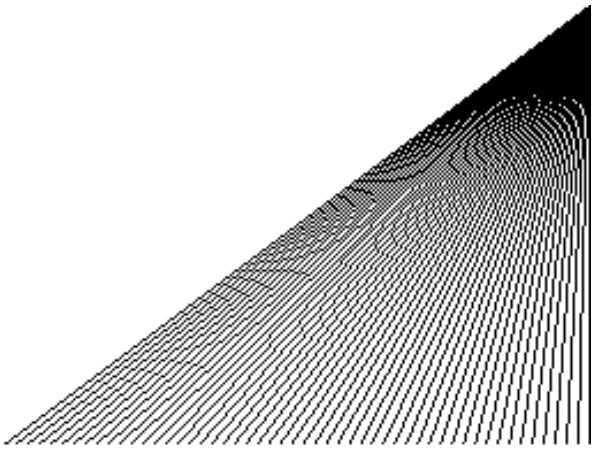
```
  10 LET HRG=32736
  20 PRINT USR HRG,ON
  30 PRINT USR HRG,CLR
 100 FOR X=0 TO 255 STEP 4
 110 PRINT USR HRG,LINE,X,0,255,
191
 120 NEXT X
 130 LIST
 140 STOP
```

When reaching line 140 the program will stop with error 9/140. The software therefore switches back to the text mode at the end.

With keys 9 + 0 you can now toggle between graphic and listing.

When you delete line 140 the program will stop showing the graphic screen without any notice. Now you have to toogle between the screens to see if the program has finished.

The program generates the following graphic:



Machine-code example

The following machine-code program does the same things as the Basic program. Each function code is expected to be in the A register. All other parameters are in the BC and DE register.

```
        ld      A, HRG_On           ; activates the HRG
        call    7FE3h               ; call the HRG
                                    ;
        ld      A, HRG_Clr          ; clears the HRG
        call    7FE3h               ; call the HRG
                                    ;
        ld      A, HRG_Line         ; draw a line between start and end
        ld      bc,0000h            ; Start x=0     y=0
        ld      de,FFBFh            ; End   x=255 y=191
                                    ;
DrawLoop:                           ;
        push    AF                  ;
        push    BC                  ; save all needed registers
        push    DE                  ;
        call    7FE3h               ; call the HRG
        pop     DE                  ;
        pop     BC                  ; recall the registers
        pop     AF                  ;
                                    ;
        inc     B                   ; modify the startpoint
        inc     B                   ; x = x+4
        inc     B                   ;
        inc     B                   ;
        jr      nz,DrawLoop         ; draw next line until x is 0 again
                                    ;
        ret                         ;
```

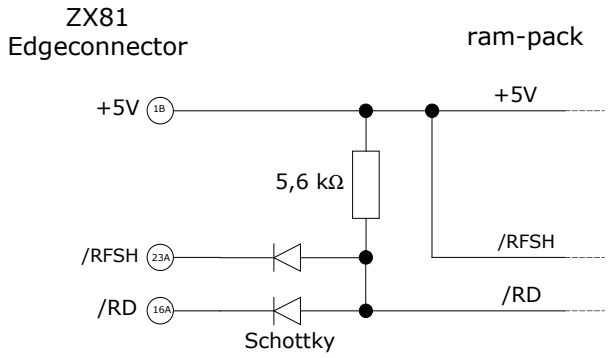
Funtion codes:

HRG_Bank	00h	HRG_Unpoly	10h
HRG_On	01h	HRG_Xpoly	11h
HRG_Off	02h	HRG_Clr	12h
HRG_Pix	03h	HRG_Text	13h
HRG_UnPix	04h	HRG_Invert	14h
HRG_XPix	05h	HRG_HCopy	15h
HRG_Line	06h	HRG_UnText	16h
HRG_Unline	07h	HRG_XText	17h
HRG_Xline	08h	HRG_OverText	18h
HRG_Lineto	09h	HRG_Circle	19h
HRG_Unlineto	0ah	HRG_Uncircle	1ah
HRG_Xlineto	0bh	HRG_Xcircle	1bh
HRG_Box	0ch	HRG_UDGnew	1ch
HRG_Unbox	0dh	HRG_UDGset	1dh
HRG_Xbox	0eh		
HRG_Poly	0fh		

How to make a ram-pack graphic capable

Nearly every ram-pack for the ZX81 or TS1000 can be made graphic capable. All you need is two diodes and a resistor.

First you have to disconnect the signals „Read" and „Refresh" (/RD and /RFSH) from the edgeconnector by cutting the track. Then connect the diodes and the resistor and a wire-bridge as shown in the following diagram.



With this modification the refresh-signal to the ram-pack will be always high or disabled. The read-signal to the ram-pack is now activ low either if the CPU generated a read or a refresh. This makes the ram-pack send the memory data to the bus when a refresh cycle happens. For normal operation of the computer or the ram-pack this is regardles. The hrg-software on the other hand just needs these data to generate a graphic-video.

Application interface

The application interface to Basic or machine-code programs is located between 7FE0 – 7FFF or 32736 – 32767.

The following memory dump shows the content:

```
7FE0   C3 40 60    JP HRG-BASIC-API
7FE3   C3 48 60    JP HRG-MC-API
7FE6   07 02       ;VER. LO HI
7FE8   2D 37 2C    ;"HRG"
7FEB   00          ;ENDMARK
```

This structure will be the same in all versions. Only the version number and the targets of the jumps will differ.

With the first jump a Basic program calls a function as shown in previous examples.

```
10 LET HRG=32736
20 PRINT USR HRG,ON
```

The second jump is for machine-code. All parameters are expected to be in Z80-registers.

```
ld      A, HRG_On          ; function code in A
call    7FE3h              ; call the HRG
```

To check if the application interface is present and the graphic software is installed the following is recommended:

1. Check if memory location 7FE0 is filled with C3 or 195.
2. Check if the two following memory-cells make a value between RAMTOP and 7FE0.

If both conditions are true it is very likely to have the graphic software installed.

If in the Basic example the interface is called with no graphic software installed you normally get an error 2/20 because the PRINT command does not find a variable with name ON.

Function description

All functions of the application interface for Basic and machine-code are desribed here.

On the left you find the function name for Basic. On the right you find the name and function number for the machine-code interface.

In the machine-code interface the function number has to be in the A-register. Most functions need the first two parameters to be given in the BC-registers and the next two parameters in the DE-registers. Functions with more parameter use either the processor stack or use a pointer to a structure of data.

A called function works only on the previously selected bank. With this scheme you can build up a graphic while viewing another graphic or a text screen.

Set, reset or invert a pixel

PIX **UNPIX** **XPIX**	**HRG_Pix** **HRG_UnPix** **HRG_Xpix**	A=03h A=04h A=05h
X	B	0 - 255
Y	C	0 - 191

Works on a pixel with position X,Y.

Draw a line with start and endpoint

LINE **UNLINE** **XLINE**	**HRG_Line** **HRG_UnLine** **HRG_XLine**	A=06h A=07h A=08h
X1	B	0 - 255
Y1	C	0 - 191
X2	D	0 - 255
Y2	E	0 - 191

Draws a line from pixel X1,Y1 to pixel X2,Y2.

Draw a line to endpoint

LINETO **UNLINETO** **XLINETO**	**HRG_Lineto** **HRG_UnLineto** **HRG_XLineto**	A=09h A=0ah A=0bh
X	B	0 - 255
Y	C	0 - 191

Draws a line from last drawn pixel to pixel X,Y.

Draw a rectangle between two pixels

BOX **UNBOX** **XBOX**	**HRG_Box** **HRG_UnBox** **HRG_XBox**	A=0ch A=0dh A=0eh
X1	B	0 - 255
Y1	C	0 - 191
X2	D	0 - 255
Y2	E	0 - 191

Draws a rectangle between pixel X1,Y1 and pixel X2,Y2.

Draw a circle with center and radius

CIRCLE **UNCIRCLE** **XCIRCLE**	**HRG_Circle** **HRG_Uncircle** **HRG_Xcircle**	A=19h A=1ah A=1bh
X	B	0 - 255
Y	C	0 - 191
R	D	0 - 255

Draws a complete circle around pixel X,Y with radius R. The circle may exceed the viewing area an any border.

Draw a polygon with multiple pixels

POLY **UNPOLY** **XPOLY**	**HRG_Poly** **HRG_UnPoly** **HRG_XPoly**	A=0fh A=10h A=11h
X1	on stack	0 - 255
Y1	on stack	0 - 191
X2	on stack	0 - 255
Y2	on stack	0 - 191
etc.	etc.	

Draws a closed polygon with tree or more corner pixels. At the end a closing line is drawn back to the fisrt pixel. In machine-code interface all parameters are to be pushed onto the processor stack before calling the function. Before that the calling programm has to push an endmark (ffh, ffh) to the stack.

Example:

```
ld      A, HRG_Poly         ; draw polygone
ld      bc,FFFFh            ; endmark onto stack
push    BC                  ;
ld      bc,0000h            ; Start x=0  y=0 onto stack
push    BC                  ;
ld      bc,0A64h            ; Pixel x=10 y=100 onto stack
push    BC                  ;
ld      bc,1432h            ; Pixel x=20 y=50 onto stack
push    BC                  ;
call    7FE3h               ; call the HRG
```

The called function pops all the data from the stack. The stack needs not to be corrected by the calling program!

Displays the graphic

ON	**HRG_On**	A=01h

Displays the previously with BANK selected graphics. It is not necessary to view a bank for drawing pixels into it. It is possible to view and draw in different banks.

Displays the text screen

OFF	**HRG_Off**	A=02h

Displays the text-screen. Drawing into graphics is unaffected by this command.

Clears a graphic

CLR	**HRG_Clr**	A=12h

The selected bank will be filled with unset pixels.

Copies the textscreen into a graphic

TEXT	**HRG_Text**	A=13h
UNTEXT	**HRG_UnText**	A=16h
XTEXT	**HRG_XText**	A=17h
OVERTEXT	**HRG_OverText**	A=18h

Makes a copy of the text in the text-screen into the selected graphic bank. It uses all previousely defined UDGs.

With TEXT all set pixels of a character are set in the graphic. With UNTEXT all set pixel of a character will be reset in the graphic. With XTEXT all set pixel of a character will be used to invert the graphic. With OVERTEXT all 64 pixel of a character (black and white) will be copied into the grapic.

Reset all user defined graphics

UDGNEW	**HRG_UDGnew**	A=1Ch

All 127 user defined characters will be set to the standard Sinclair characters.

Define a character

UDGSET	**HRG_UDGset**	A=1Dh
C	BC points to	0 - 128
B0	a structure of	0 - 256
B1	data	0 - 256
B2		0 - 256
B3		0 - 256
B4		0 - 256
B5		0 - 256
B6		0 - 256
B7		0 - 256

The bitpattern for character C is set with bytes B0 to B7. In Sinclair Basic codes 0 to 63 are normal characters and codes 128 to 191 are inverse characters. With this function all normal and inverse characters can be defined independently. So you have 127 characters that can be redifined. If C has a value between 64 and

127 or between 192 and 255 then the function interpretes the value as if it was betwen 128 and 191.

In the following example the Pound character is substituted with a Euro-character.

Example in Assembler:

```
        ld      A, HRG_UDGset       ; set a character
        ld      bc, MyEuro          ; poiter to a structure of 9 bytes
        call    7FE3h               ; call the HRG
        jp      ...                 ;
MyEuro:                             ;
        db      12                  ; character code for „£"
        db      0, 60, 66, 248      ; graphic bits for a „€" character
        db      64, 248, 66, 60     ; 0     = 00000000
                                    ; 60    = 00111100
                                    ; 66    = 01000010
                                    ; 248   = 11111000
                                    ; 64    = 01000000
                                    ; 248   = 11111000
                                    ; 66    = 01000010
                                    ; 60    = 00111100
```

Example in Basic:

```
  10 LET HRG=32736
  20 PRINT USR HRG,UDGSET,12,0,6
0,66,248,64,248,66,60
  30 PRINT "123£"
  40 PRINT USR HRG,TEXT
  50 PRINT USR HRG,ON
```

Invert the graphic

INVERT	**HRG_Invert**	A=14h

Inverts all pixels in the selected bank.

Print a graphic

HCOPY	**HRG_Hcopy**	A=15h

Prints the selected graphic on a Sinclair compatible printer.

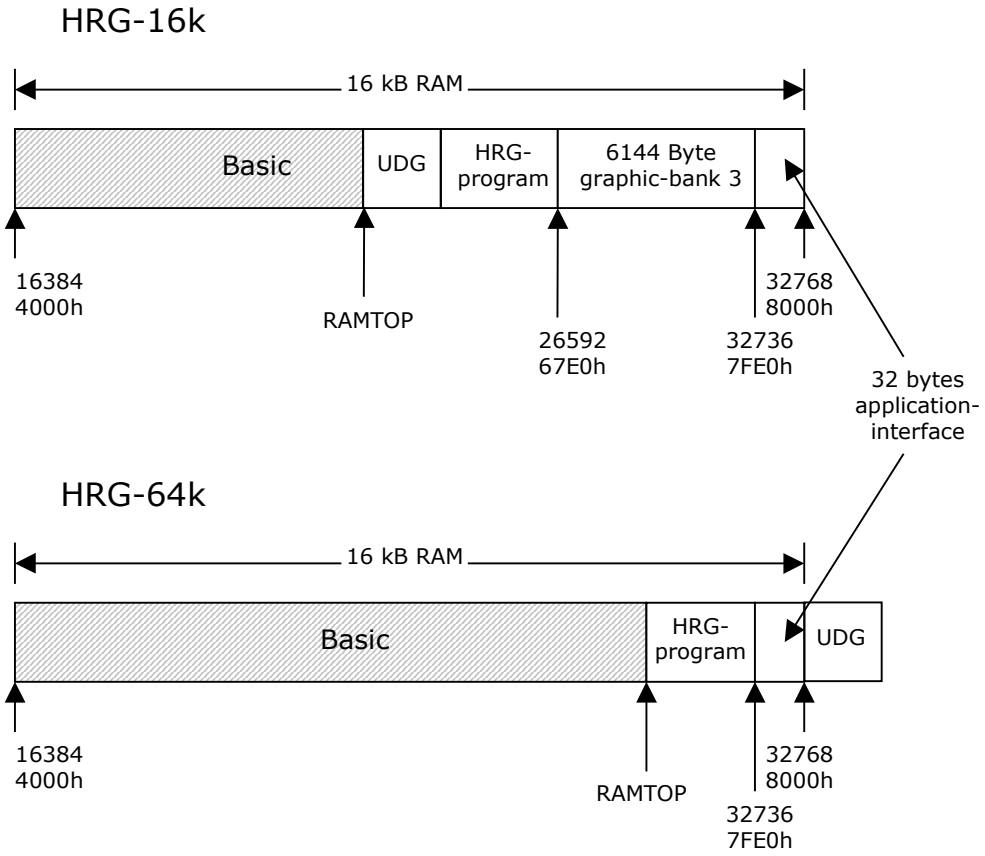
Select bank

BANK	**HRG_Bank**	A=00h
N	BC	3 –7 or ffffh

Selects a bank. In machine-code you have the option to querry the actually selected bank and the starting address of the graphic. To do this you have to load BC with ffffh. On return the bank number is in the A-register and the starting address of the graphic is in the HL-Registers.

Memory map

The following diagram shows the locations of the program and all the graphic memory addresses.



On the 64k version there is no bank 3. So you keep more memory for the Basic program.

Graphic-bank 3	67E0h to 7FE0h -1
Graphic-bank 4	87E0h to 9FE0h -1
Graphic-bank 5	A7E0h to BFE0h -1
Graphic-bank 6	C7E0h to DFE0h -1
Graphic-bank 7	E7E0h to FFE0h -1