**CREATIVE COMPUTER**



MONITOR HANDLING  
MANUAL



First Edition

August, 1960

SORD COMPUTER CORPORATION

1. This document may not be reproduced in any form, in whole or in part, without the express written permission of SORD COMPUTER CORPORATION.
2. The contents of this manual may be changed without prior notice.
3. This manual has been carefully prepared and reviewed for completeness and accuracy. However, contact us concerning any errors which you may notice or on any points on which you are in doubt.
4. SORD COMPUTER CORPORATION, assumes no responsibility for any consequences which may arise from the use of this manual or for any errors which it may contain.

TABLE OF CONTENTS

[Chapter 1 Basic concept of M5 Monitor System 1](#bookmark1)

Chapter 2 Display handler 2

[Chapter 3 Keyboard handling 3](#bookmark42)

[Chapter 4 Sprite handler 4](#bookmark54)

[Chapter 5 Event handler 5](#bookmark59)

Chapter 6 Saving to and loading from cassette

storage 6

[Chapter 7 Sound generator handler 7](#bookmark81)

[Chapter 8 Printer handler 8](#bookmark94)

[Chapter 9 Other System Information 9](#bookmark98)

Chapter 10 Appendices 10

Chapter 1 Basic Concept of M5 Monitor System

The M5 computer belongs to the inexpensive class of personal computer. Most personal computers of this class have been hobbyist or introductory machines, probably because of the makers' cost factor. Many of these machines use a tiny BASIC in ROM. At first glance, this seems reasonable but this arrangement restricts the flexibility of machines.

This system monitor does not provide any of the so-called OS features. Contents of the monitor ROM (8KB) are all drivers routines and are written as subroutines. Users can freely access software in the ROM and can utilize monitor subroutines in their own application programs. About 200 monitor subroutines are available. Nearly all subroutines the user will need are included with the system.

This computer has, as a system table, more than 500 bytes of RAM, the values of which are initialized at bootstrapping and can be redefined by the user. Any system can be set by the user, from the video RAM memory mapping to the keyboard auto repeat interval. By replacing the application RAM cartridge, this computer can function as a BASIC machine, another-language machine, or a communication terminal.

Thus, users who are engaged in software development should become familiar with the internal subroutines and the system table.

CONTENTS

1. [Function of VDP TMS9918A (LSI) 2-1](#bookmark2)
2. [VDP register set 2-2](#bookmark3)
3. [Four display modes of VDP 2-7](#bookmark6)
4. [Processing screen and spare screen 2-15](#bookmark16)
5. VRAM configuration 2-19
6. [Viewport 2-20](#bookmark19)
7. [Character definition 2-23](#bookmark21)
8. [Color definition 2-27](#bookmark23)
9. [One-character display routine 2-34](#bookmark25)
10. [How to handle control codes 2-36](#bookmark26)
11. [Two modes for one-character display 2-44](#bookmark28)

(Overwrite and insert)

1. [Character read-on screen 2-46](#bookmark29)
2. [One-line display routine 2-47](#bookmark30)
3. [One-line read routine 2-50](#bookmark31)
4. Cursor move and cursor out 2-51
5. [Scrolling and shifting 2-55](#bookmark33)
6. [Matrix display and read 2-60](#bookmark34)
7. Sentence definition 2-63
8. [Search for sentence top and next sentence top 2-63](#bookmark37)
9. [Transfer instructions 2-67](#bookmark39)
10. [Multiplication routine 2-73](#bookmark40)
11. [Other information 2-74](#bookmark41)
12. 1 Function of VDP TMS9918A (LSI)

2-1-1 Display mode

The TMS9918A provides the following four display modes:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Mode** | **Resolution (in dots)** | **Pattern**  **size**  **(in dots)** | **No. of patterns** | **Color**  **assignment** | **Sprite** | **Display**  **screen** |
| **Graphics**  **I** | **192 x 256** | **8x8** | **256** | **16 colors** | **Usable** | **24 rows x 32 columns** |
| **Graphics**  **II** | **192 x 256** | **8x8** | **768** | **16 colors** | **Usable** | **it** |
| **Multi**  **color** | **48 x 64** | **1 block 4x4** | **-** | **16 colors** | **Usable** | **ii** |
| **Text** | **196 x 256** | **KD**  X  00 | **256** | **2 colors** | **Not**  **usable** | **24 rows x 40 columns** |

2-1-2 Sprite

The TMS9918A can output 32 animation patterns, called sprites. A sprite can change the screen freely both horizontally and vertically in dot increments and is not restricted by the character screen.

A sprite is characterized by the following items.

1. Each sprite can move freely within the display screen by changing its coordinates. It does not require pattern redefinition.
2. The 32 sprites have assigned priorities. If two or more sprites overlap, the overlapping part of the lower-priority sprite is erased automatically. Hence, three-dimensional expression is easily

available. Users are not required to handle the processing of the shaded part. The sprite priority is ranked from the highest sprite (#0) to the lowest (#31).

1. Up to 256 kinds of patterns of sprite can be established, out of which 32 can be displayed.
2. A sprite consists of sprite patterns, each of which can be magnified by two times.
3. Any one of sixteen sprite colors can be specified.
4. The VDP status is established at one-screen scan completion time.
5. 2 VDP register set

The TMS9918A (called VDP) consists of eight write-only registers and one read-only register. The screen display takes place by setting data in the VDP register and write image data in the VRAM (video RAM).

Register **#0**, Register **#1** Register #0

76543210

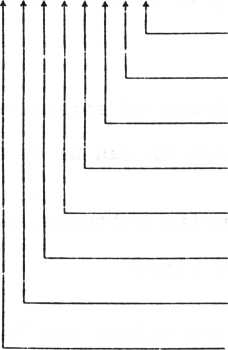
^ External video signal

Mode selection bit (M3)

Register *#1*

**76543210**

MAG: Sprite magnification



SIZE: Sprite size Set to 0.

Mode selection bit (M2)

Mode selection bit (Ml)

IE: Interrupt enable

BLANK: Screen display ON/OFF

VRAM selection

External video signal 0: Input disabled

1: Input enabled

Display mode selection

|  |  |  |  |
| --- | --- | --- | --- |
| **Ml** | **M2** | **M3** | **Display mode** |
| **0** | **0** | **0** | **Graphics I** |
| **0** | **0** | **1** | **Graphics II** |
| **0** | **1** | **0** | **Multi-color** |
| **1** | **0** | **0** | **Text** |

Sprite magnification

0: Does not magnify (1:1 display)

1: Magnified by 4.

Sprite size

0: 8 dots x 8 dots

1: 16 dots x 16 dots

. Interrupt enable

0: CPU interrupt disabled.

1: CPU interrupt enabled.

. BLANK

0: Screen erase

1: Screen display

. VRAM selection 0: 4K

1: 8K or 16K

Register **#2**

**76543210**

**0**



**Sets the higher 4 bits of the pattern name table head address.**

Register **#3**

76543210

Sets the higher 8 bits of the character color table head address.

Register #4

**76543210**

L

**Sets the higher 3 bits of the character generator table head address.**

**76543210**

Register 6

**Sets the higher 7 bits of the sprite attribute table head address**

**76543210**



**Sets the higher 3 bits of the sprite pattern generator table head address.**

**Register #1**

**76543210**

Text color "0" Backdrop color

Text color "1"

• Text color "0", backdrop color

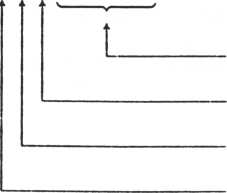
Color of the part where each character bit is "0" in the text mode, and the backdrop color. In modes other than the text mode the backdrop color is given.

. Text color "1"

Color of the part where each character bit is "1" in the text mode.

**76543210**

**5th sprite no. Coincidence flag 5th sprite Interrupt flag**



. Interrupt flag

When the IE (interrupt enable) bit in this register is set to "l", VDP INT is output after a one-screen scan is completed.

. 5th sprite (5S) and 5th sprite no.

When more than five sprites are on the same horizontal line and the interrupt flag (F) is 0, the fifth sprite bit is set to "1". The fifth sprite number is maintained in the lower 5 bits.

. Coincidence flag

When a screen has two or more coincidence sprites, this flag is set to "1".

The system reads the status for each frame, and therefore, users are not required to read the status.

Subroutine calling sequence

VREGI

**Subroutine name: Address:**

**Function: Register save:**

OCA3H

Initializes the VDP registers. X, Y

The VREGI routine sets the VDP register according to the processing screen information table.

The call takes place after necessary information has been written into the processing screen information table.

1. 3 Four display modes of VDP

The VDP TMS9918A has four display modes. The characteristics and procedure for establishing each mode are detailed below.

2-3-1 Graphics I mode (GI mode)

In the graphics I mode 8x8 dot characters can be displayed in 24 rows and 32 columns. A maximum of 256 characters can be defined.

For the character colors, one of the colors of the part where the bit status is "1" can be specified for 8 characters, and one color of a part in the bit "0" status. This means that 32 kinds of different colors can be displayed per 256 characters.

The graphics I mode is the standard mode of this computer.

2-3-2 Graphics II mode (GII mode)

In the graphics II, or GII, mode full graphics are available, and 768 characters can be defined for all of the 768 character positions in 24 rows and 32 columns.

Hence, a particular pattern different from other patterns can be defined for a part where the screen appears.

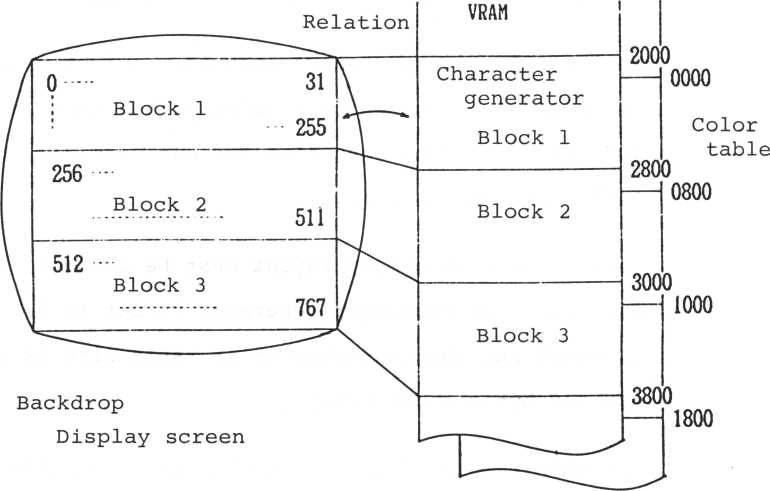
In the GII mode two colors (the color of the part in the bit "1" status and the color of the part in the bit "0" status) can be specified for one vertical dot and eight horizontal dots.

Hence, the VRAM memory layout must be changed in the GII mode: the character generator is set to 6KB (2KB in GI mode) and the character color table size is set to 6KB (32 bytes in GI mode).

The VRAM memory layout in the GII mode is called layout II (in other modes layout I) for distinguishing these two modes (refer to Attachment: Memory Map).

Both the character generator and character color table occupy 6KB. This means that a screen can be split into three blocks (upper, middle and lower), and that a character generator, character color, and pattern name table can be established for each block.

Backdrop



2-3-3 Multi-color mode

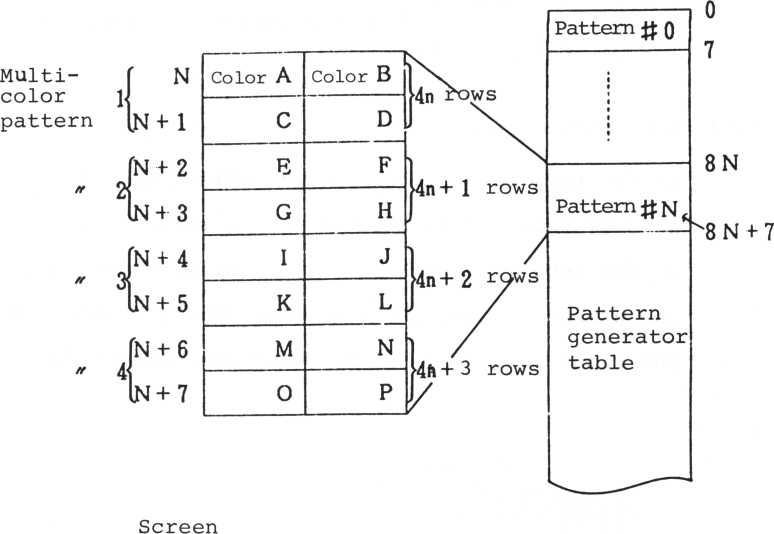
In the multi-color mode color blocks in increments of

4 dots x 4 dots are arranged in 48 rows and 64 columns. Each color block pattern is established by the pattern generator table. A pattern with four vertical characters (16 blocks) is established from every eight bytes counted from the head of the pattern generator table.

VRAM

**Relative**

**address**



**Row p**

1

2

22

23

**Pattern**

N

(1)

**Pattern**

N

(**2**)

**Pattern**

N

**(3)**

Color patterns are displayed on the screen as follows.

Establishing the pattern N to (1) row 0 (4n rows) causes

|  |  |
| --- | --- |
| **Color A** | **Color B** |
| **C** | **D** |

**of the pattern N to be displayed.**

Establishing the pattern N to (2) row 2 (4n + 2) causes

of the pattern N to be displayed.

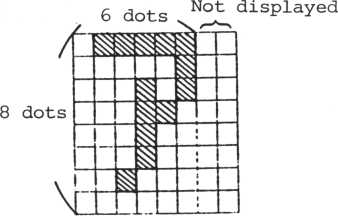
|  |  |
| --- | --- |
| **Color I** | **Color J** |
| **K** | **L** |

2-3-4 Text mode

|  |  |
| --- | --- |
| **Color M** | **Color N** |
| **0** | **P** |

**of the pattern N to be displayed.**

In the text mode the character pattern side is split into 24 rows and 40 columns on the screen. However, as the resolution of the entire screen does not change, the number of dots per character is 6 x 8 dots. Hence, two right dots of each character are not displayed.



Up to 256 characters can be established, and two colors can be specified per entire screen: a color for the part in the bit 1 status and a color for the part in the bit 0 status. In the text mode it is handled as a premise not to display a graphic pattern.

2-3-5 Establishing a display mode

A mode is established by initializing the processing screen information table within the system table and resetting the VDP register.

GI mode establishment Subroutine calling sequence

|  |  |
| --- | --- |
| **Subroutine name:** | **CMODE** |
| **Address:** | **0DD8H** |
| **Function:** | **Establishes the GI mode.** |
| **Register save:** | **X,Y** |
| **GII mode** |  |

Subroutine calling sequence

|  |  |
| --- | --- |
| **Subroutine name:** | **GMODE** |
| **Address:** | **0B1FH** |
| **Function:** | **Establishes the GII mode** |
| **Register save:** | **X,Y** |
| **Other conditions:** | **Layout error**  **(Other screen is already in the** GII **mode or multi-color mode.)** |
| **T (text) mode** |  |

Subroutine calling sequence

|  |  |
| --- | --- |
| **Subroutine name:** | **TMODE** |
| **Address:** | **0D04H** |
| **Function:** | **Establishes the text mode.** |
| **Register save:** | **X,Y** |

**Subroutine calling sequence**

**Subroutine name:**

**MMODE**

**Address:**

**OC44H**

**Function:**

**Establishes the multi-color mode.**

**Register save:**

**X,Y**

**Other conditions:**

**C = 1 .. Layout error**

**(Other screen is already in the GII mode.)**

**Precaution**

**In every mode, character generators and color tables are initialized by a special routine. Hence, establishing a mode after a color has been set by a user destroys characters and color data defined already.**

**2-3-7 Precautions for establishing a mode**

**(1) Graphics I mode**

**When the processing screen mode is set to the graphic I mode, the following processing takes place:**

**a. If the preceding screen is in the graphics I**

**mode, no operation takes place.**

b. In other modes, the screen and sprite on the screen are cleared, the character generator is initialized, then the color table is initialized. The initial value of the color table shows characters in grey and the background in black.

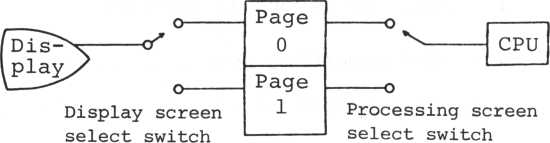
1. Graphics II mode
2. When the spare screen is in the graphics II or multi-color mode, a layout error occurs.
3. When the preceding screen is in the graphics II mode, no operation takes place.
4. In other modes, the screen and sprite on the screen are cleared, then the generator and color table are initialized.
5. Text mode
6. When the preceding screen is in the text mode, no operation takes place.
7. When the preceding screen is in the graphics II mode, the layout is changed.
8. When the preceding screen is not in the text mode, the screen is cleared and the character generator is initialized.
9. When the spare screen is in the graphics II mode, a layout error occurs.
10. The screen and sprite are cleared.

2-4 Processing screen and spare screen

This computer is furnished with 16KB VRAM and has two screen in VRAM. The screen with the smaller number of addresses is called page 0, and the other page 1. On the other hand, in the aspect of the CPU processing, the screen in processing is called a processing screen, the other a spare screen.

The processing screen is capable of freely reading and rewriting VRAM data. In contrast, the spare screen cannot convert nor read VRAM data. But, on both the processing screen and spare screen the display can be optionally switched.

**VRAM**



(The screen connected to this switch is called the display screen.)

(The screen connected to this

switch is called the processing screen, the other the spare screen.)

in the system table.

SVSSSW

**76543210**

— Processing screen (0: page 0, 1: page 1)

— Display screen (0: page 0, 1: page 1)

— The screen that is displayed -

0: The processing screen 1: The spare screen

This computer system has two screen information tables in the system table. One is the processing screen information table and the other is the spare screen information table.

Switching the processing screen between page 0 and page 1 is done by exchanging contents of the two screen information tables.

Switching of the display screen is operated by displaying one of the two screen information tables.

Screen switching routines and precautions

Processing screen selection

Subroutine calling sequence

Subroutine name: WRTSC

|  |  |
| --- | --- |
| **Address:** | **OE33H** |
| **Function:** | **Selects the processing screen** |
| **Input**  **Registers** | **Contents** |
| **Acc** | **n n = 0 Selects page 0.** |
|  | **n = 1 Selects page 1.** |
|  | **n > 1 Reverses a page.** |

Register save: C,X,Y

Display screen selection Subroutine calling sequence

|  |  |
| --- | --- |
| **Subroutine** | **name: DSPSC** |
| **Address:** | **OC64H** |
| **Function:** | **Selects the display screen.** |
| **Input**  **Registers** | **Contents** |
| **Acc** | **n n = 0 Displays page 0.** |
|  | **n = 1 Displays page 1.** |
|  | **n > 1 Reverses a display page.** |

Register save: X,Y

Subroutine calling sequence

FRMSC

**Subroutine name: Address:**

**Function:**

OC77H

Selects the display screen,

**Input**

**Registers**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Acc** | **n** | **n =** | **0** | **Displays** | **the** | **processing** |
|  |  |  |  | **screen.** |  |  |
|  |  | **n =** | **1** | **Displays** | **the** | **spare screen.** |
|  |  | **n >** | **1** | **Reverses** | **the** | **screen.** |

Contents

Subroutine calling sequence

NRMSC

**Subroutine name: Address:**

**Function:**

**Register save:**

0C5CH

Sets both the display screen and processing screen to page 0.

X,Y

Display screen change

Subroutine calling sequence

Subroutine name: RVDSPP

Address: OC62H

Function: Reverses the display screen.

Register save: X,Y

Processing screen change

Subroutine calling sequence

Subroutine name: RVWRTP

Address: 0E31H

Function: Reverses the processing screen,

Register save: B,C

Display screen and processing screen change

Subroutine calling sequence

Subroutine name: REVSC

Address: 0E2EH

Function:

**Reverses both the display screen and processing screen,**

**X, Y**

Register save:

2-5 VRAM configuration (VRAM layout)

This system has four display modes: GI, GII, M, and T. The GI, M and T modes require a VRAM capacity of about 4KB for one screen. The GII mode requires a VRAM capacity of about 13KB for one screen. Hence, the VRAM layout depends on whether the GII mode is used or not.

(See Attachment: Memory Map)

Layout I: When the GII mode is not used, the 8 KB of VRAM between 0000H and 1FFFH are unused and released for the users.

Layout II: When the GII mode is used, the entire VRAM is displayed on the screen.

The layout is switched automatically by the system when a user calls the subroutine to set a mode. Hence, no action required by the users.

There are two points of which the user should be aware:

1. In layout 1, the area of 8KB from 0000H on the VRAM is unused but layout II uses this area, so data is destroyed if the layout is switched when a user stores data in this area.
2. In layout II, three tables (sprite pattern generator table in the GII mode, and sprite pattern generator and character generator of other screens) are defined in the same area and overlap in memory. Hence, if the sprite is defined in the GII mode and the display is switched to the other screen, characters on the screen may be destroyed.

2-6 Viewport

The viewport is a small screen that is logically established inside of the physical screen. Screen control functions can be performed only on the viewport, leaving the rest of the screen unchanged.

Establishing a viewport allows various screen control to be performed only within the viewport; the screen outside the viewport remains unchanged.

Definition of left margin, upper margin, height and width of vewport

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scree]**  **heighi**  j | r  1  t  i |  | **View**  **port** |  |
|  | | **<— Screen —•» width** | | |

0

**(T mode)**

23

|  |  |  |  |
| --- | --- | --- | --- |
| **1 Upper margin** | | |  |
| **Left**  **margin** | **J ( Xi ,Yi)**  **Width** | **Height** | **Right**  **margin**  **f. .** |
|  | **(X2 .Y2**  **ri** |
|  | **Lower margin j** | | **;** |

**Y**

**direc**

**tion**

**X direction**

0 ► 3i or 39

Establishing a viewport requires a left margin, upper margin, right margin, and lower margin. Each parameter is given by a coordinate in character units.

Let the character coordinate of the upper left corner of the viewport be (X^, Y^) and the character coordinate of the lower right corner be (X2, Y2). Then

Left margin of viewport =

Upper margin of viewport = Right margin of viewport = Lower margin of viewport =

Viewport establishment

Subroutine calling sequence

Subroutine name: VIEWP

|  |  |
| --- | --- |
| **Address:** | **135CH** |
| **Function:** | **Establishes a viewport.** |
| **Input**  **Registers** | **Contents** |
| **H** | **Left margin** |
| **L** | **Right margin** |
| **D** | **Upper margin** |
| **E** | **Lower margin** |

Register save: X,Y

Other conditions: Cy = 1 Range error

Tried to establish a viewport larger than the screen.

Establishing a viewport larger than the screen causes an error.

VIEWRS

1353H

**Subroutine name: Address:**

**Function: Register save:**

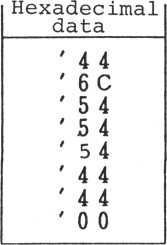
Releases the viewport. X,Y

Changing a mode during the viewport establishment releases the viewport automatically.

2-7 Character definition

Binary 8-byte image data is handled as one character.

7 6 5 4 3 2 1 0



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | r |  | A | 1  **i** |  |  |
|  |  |  |  |  |  |  |  |

**Subroutine calling sequence**

**Subroutine name: STCHR**

**Adress: OE59H**

**Function: Establishes character patterns,**

**sprite patterns, and character colors in the GII mode.**

**Input**

**Registers**

**Acc**

**Contents**

**n (character set no.) n = 0: Sprite pattern 1: Character set 1**

**B**

**C**

**2: " 2**

**3: " 3**

**4: Color set 1**

**5: " 2**

**6: " 3**

**No. of characters**

**Establishment start code no.**

**H,L:**

**Data head address**

**Register save: X, Y**

**Other conditions: Cy = 1 Parameter inadequqte**

**(Value over 7 set in Acc, etc,) Mode error**

**Values of 2 to 6 were set in the Acc in modes other than GII.**

|  |  |
| --- | --- |
| Acc: n = 0 | Sprite pattern establishment |
| 3  ii | Character pattern established (all modes) |
| n = 2 | Block 2 character established in the GII mode. |
| n = 3 | Block 3 character established in the GII mode. |
| **ii**  C | Block 1 color established in the GII mode |
| n = 5 | Block 2 color established in the GII mode |
| n = 6 | Block 3 color established in the GII mode |
| B: Number | of characters to be established. |

C: Code no. of a character head to be determined.

H,L: Data head address

Character pattern establishment

Subroutine calling sequence

Subroutine name: STCCHR

|  |  |
| --- | --- |
| Address: | OE55H |
| Function: | Establishes characters in modes other than the GII mode.  (When Acc is set to 1 by STCHR) |
| Input  Registers | Contents |
| B | Number of characters to be established. |
| C | Establishment start code no. |
| H, L | Data head address |

Register save: X,Y

Area used in modes other than the GII mode.

Subroutine calling sequence

Subroutine name: LCPAT

Address: OD89H

Function: Initializes character patterns.

Register save: X,Y

This routine initializes the entire VRAM character pattern generator the default status. For the default character pattern, see the attachment.

Character pattern read out

Subroutine calling sequence

Subroutine name: RDCHR

Address: OE75H

Function: Reads out character patterns,

sprite patterns and color data

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | in the GII | mode. |
| Input |  |  |  |
| Registers | Contents | |  |
| Acc | n |  |  |
|  | n = 0: | Strite |  |
|  | 1: | Character | set 1 |
|  | 2: | it | 2 |
|  | **3:** | ii | 3 |
|  | 4: | Color set | 1 |
|  | 5: | **ii** | 2 |
|  | 6: | **ii** | 3 |

C Read-out start code no.

H,L Read-in buffer head address

Register save: X,Y

Other conditions: C =1 ••• Parameters inadequate.

(When more than 7 parameters are set in Acc)

Mode error

(In modes other than the GII mode, tried to read out in the GII mode.)

"RDCHR" is processed in the reverse procedure to "STCHR" and reads into main memory such data as character patterns, color patterns, etc. already

set in VRAM.

2-8 Color definition

Colors defined by this computer can be of three kinds: character color, background color, and drop-out color. The three kinds of colors are specified by one of 16 color codes.

"Transparent color" allows the background color to be seen.

2-8-1 Establishment of character colors and background colors in graphics I mode

|  |  |
| --- | --- |
| **Color** | **Code** |
| **Transparent** | **0** |
| **Black** | **1** |
| **Green** | **2** |
| **Light green** | **3** |
| **Dark blue** | **4** |
| **Light blue** | **5** |
| **Dark red** | **6** |
| **Cyan** | **7** |
| **Red** | **8** |
| **Light red** | **9** |
| **Dark yellow** | **A** |
| **Light yellow** | **B** |
| **Dark green** | **C** |
| **Magenta** | **D** |
| **Grey** | **E** |
| **White** | **F** |

In the graphics I mode, a character color and a background color can be set for every eight characters. Hence, a character is split into 32 blocks (listed below) to establish the block color.

Subroutine calling sequence

|  |  |  |  |
| --- | --- | --- | --- |
| Block  No. | Code No. | Block  No. | Code No. |
| 0 | 0 - 7 (' 0-' 7) | 16 | 128 - 135 (' 80- ' 87) |
| 1 | 8 — 15 (' 8~' F) | 17 | 136 - 143 (' 88- ' 8F) |
| 2 | 16 — 23 (' 10— ' 17) | 18 | 144 - 151 (' 90- ' 97) |
| 3 | 24 — 31 (' 18- ' IF) | 19 | 152 - 159 (' **98-' 9F)** |
| **4** | **32 — 39 (' 20— ' 27)** | **20** | **160 — 167 (' A0-' A7)** |
| **5** | **40 — 47 (' 28~ ' 2F)** | **21** | **168 - 175 (' A8- ' AF)** |
| **6** | **48 — 55 (' 30— ' 37)** | **22** | **176 - 183 (' B0- ' B7)** |
| **7** | **56 ~ 63 (' 38— ' 3F)** | **23** | **184 - 191 (' B8- ' BF)** |
| **8** | **64 — 71 (' 40- ' 47)** | **24** | **192 - 199 (' CO- ' C7)** |
| **9** | **72 — 79 (' 48- ' 4F)** | **25** | **200 - 207 (' C8- ' CF)** |
| **10** | **80 — 87 (' 50— ' 57)** | **26** | **208 - 215 (' DO- ' D7)** |
| **11** | **88 — 95 (' 58— ' 5F)** | **27** | **216 - 223 (' D8- ' DF)** |
| **12** | **96 — 103 (' 60- ' 67)** | **28** | **224 - 231 (' E0-' E7)** |
| **13** | **104 - 111 (' 68-' 6F)** | **29** | **232 - 239 (' E8- ' EF)** |
| **14** | **112 - 119 (' 70-' 77)** | **30** | **240 - 247 (' F0- ' F7)** |
| **15** | **120 - 127 <' 78- ' 7F)** | **31** | **248 - 255 (' F8- ' FF)** |

**Out of 8 bits for the character colors and background colors, the higher 4 bits determine the character color and the lower 4 bits determine the background color.**

Subroutine name: STICOL

0ED3H

Address: Function:

Establishes character colors and background colors in the GI mode.

Input

Registers Contents

B Code no.

C Color data

Register save: B,C,X,Y

Other conditions: Mode error

(In modes other than the GI mode)

Color data structure

Remarks:

T i 1 r i .

1 1 ! I I \*

Character color Background color data

Subroutine calling sequence

Subroutine name: STCTBL

|  |  |
| --- | --- |
| Address: | ODF8H |
| Function: | Initializes the color table |
|  | in the GI mode. |

Register save: C,D,E,H,L,X,Y

|  |  |
| --- | --- |
| Remarks: | The initialized data uses |
|  | values stored in BDCOLA |
|  | (709FH) of the screen |
|  | information table. |

"STCBL" sets the character color to grey and the back ground color to black at the color table initialization in the GI mode.

Subroutine calling sequence Subroutine name: ITG2M

|  |  |
| --- | --- |
| Address: | ODFBH |
| Function: | Initializes color tables |
|  | for the GII mode. |
| Input  Registers | Contents |
| B ,C | Number of output bytes (In this case, store 1800H in the |
|  | B and C registers.) |

Register save: E,H,L

"ITG2M" sets the character color to grey and background color to black at the color table initialization in the GII mode.

2-8-2 Color establishment in graphics II mode

In the GII mode, two colors can be specified for each byte of the 8x8 pattern. This means that eight bytes of data are needed to specify the color of one character. (sS^i

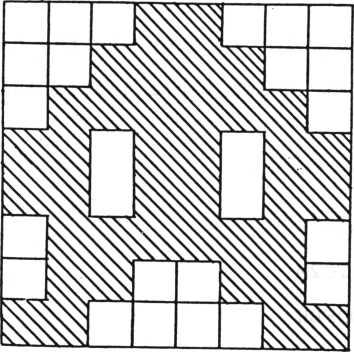
**Bit**

□

**Bit = "0"**

7 6 5 4 3 2 1 0

|  |  |  |
| --- | --- | --- |
|  |  | **Color** |
| **color** | **i. i**  **color** | **data** |
| **Green** | **Black** | **21H** |
| **Green** | **Black** | **21H** |
| **Green** | **Black** | **21H** |
| **Green** | **Red** | **28H** |
| **Green** | **Red** | **28H** |
| **Green** | **Black** | **21H** |
| **Green** | **Black** | **21H** |
| **Yellow** | **Black** | **B1H** |



**Sets 4, 5, and 6 in Acc**

**Subrouting calling sequence**

**Subroutine name: STCHR**

**Address: OE59H**

**Function: Establishes character patterns,**

**sprite patterns and characters the GII mode.**

**Input**

**Registers**

Acc

**Contents**

**N (character set no.) n = 0: Sprite pattern 1: Sprite pattern 1**

**2: " 2**

**3: " 3**

**4: Color set 1**

**5: " 2**

**6: ” 3**

B

**C**

**H,L**

**Number of characters Establishment start code no. Data head address**

**Register save: X,Y**

**Other conditions: Cy=l...Parameters inadequate**

**(Values over 7 are set in Accf etc.)**

**Mode error**

**(Tried to establish in GII mode, although not in the GII mode.)**

In the text mode two colors, the character color and the background color, are specified for each screen. (In the T mode the background color and backdrop color are always the same.)

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine | name: STFCOL |
| Address: | OC83H |
| Function: | Changes character colors in the text mode. |
| Input  Registers | Contents |
| B | Establishes a character color by the lower 4 bits. |

Register save: D,E,X,Y

|  |  |
| --- | --- |
| Remarks: | When the display mode is not the text mode, rewrites a value (higher 4 bits of BDCOLA) the screen information table and terminales. In the text mode, changes the character color. |

Subroutine calling sequence

Subroutine name: STBCOL

Address: OC97H

Function: Establishes boundary colors.

**Input**

**Registers Contents**

B Boundary color (establishes to

lower 4 bits)

Register save: D,E,H,L,X,Y

Remarks: Establishes a boundary color

without regard to the screen mode.

2-9 One-character display routine

The procedure required for displaying characters on the display screen is only the write character code nos. to be displayed in the position specified by the VRAM pattern name table. The control codes (00H - 1FH) are usually not handled as characters, but implement processing specified by the control code. On the other hand, it is possible to display a control code as a character by switching the\*display mode.

Basic one-character display processing

Subrouting calling sequence

Subroutine name: DSPCH

|  |  |
| --- | --- |
| Address: | 1088H |
| Function: | Displays one \* character. |
| Input  Registers | Contents |
| Acc | Character code to be displayed. |
| Register save: | Acc, PSW, B, C, D, E, H, L, X, Y |
| Remarks: | Displays one character in the |
|  | current cursor position and |
|  | moves the cursor to the next |
|  | position. |

DSPCH changes the processing status by the DIFLGA status of the processing screen information.

Subroutine calling sequence

Subroutine name: DSPCHA

|  |  |
| --- | --- |
| Address: | 1083H |
| Function: | Displays one character |
| Input  Registers | Contents |
| Acc | Display character code |
| Register save: | Acc, PSW,C,D,E,H,L,X,Y |
| Remarks: | The control code is executed. |

DESPCHB

1082H

**Subroutine name: Address:**

**Function:**

Displays one character,

**Contents**

Input

Registers

**Acc**

Register save: Remarks:

Display character code

Acc, PSW,C,D,E,H,L,X,Y

Control codes are handled as characters.

2-10 How to handle control codes I

Kinds and processing of each control code are listed below. (The control codes are executed by calling the control code subroutines or using DSPCHA.)

Code 00H Ignore: Regards as a terminator of the

line end.

01H Ignore

02H Searches the sentence head.

03H Lower scroll

04H Left scroll

05H Upper scroll

06H Right scroll

07H Bell

08H Delete

09H Tabulation

OAH Line feed

|  |  |
| --- | --- |
| Code OBH | Home position  (Moves the cursor to the upper left corner of the viewport.) |
| " OCH | Clear screen |
| " ODH | Carriage return |
| " OEH | Searches for the next sentence head. |
| " OFH | Overwrite mode |
| " 10H | Insert mode |
| " 11H | Multi-color mode |
| " 12H | Graphics 1 mode |
| " 13H | Graphics II mode |
| " 14H | Text mode |
| " 15H | Sets both the display screen and processing screen to page 0. |
| " 16H | Switches both the display screen and process ing screen. |
| " 17H | Carriage return |
| " 18H | Cancel |
| " 19H | Display page switching |
| " 1AH | Processing page switching |
| " 1BH | No operation |
| " 1CH | Moves the cursor right by one character. |
| " 1DH | Moves the cursor left by one character. |
| " 1EH | Moves the cursor up by one character. |
| " 1FH | Moves the cursor down by one character. |

Control codes not detailed in other sections are described below.

BELL

A single sound is output by a sound generator for a fixed time.

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine name: | BEL |
| Address: | 1176H |
| Function: | Outputs the bell sounds. |
| Register save: | Acc PSW, C,D,E,H,L |

The duration and pitch of MBEL" are set by values in the processing screen information table.

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine name: | BELK |
| Address: | 116AH |
| Function: | Outputs the keyboard click |

Register save: Acc, PSW, C,D,E,L

The keyboard click is processed, and duration and pitch of "BELK" are set by values in the keyboard information table.

DELETE

Erases one of characters in the cursor position. Unless there is a character in the cursor position, erases the preceding character. Unless there is a character in the cursor position when the cursor is positioned in the line head, nothing happens.

After the character has been deleted, the cursor returns to the preceding position. If the cursor position is followed by characters, the characters are moved forward one by one. In this case, the cursor position remains unchanged.

Subroutine calling sequence

Subroutine name: DELTC

Address: OFABB

Function: Deletes one character.

Register save: Acc, PSW, B,C,D,H

. TABULATION

Partitions the screen into eight-character zones. Others tabs can be set. When the viewport is cut, eight-character tabs are set within the viewport.

0246024602460 2 460

□Tab^ ^

Tab

□

Tab

**\***

**Without viewport**

024602460

□ Tab Tab

With viewport

1158H

**Tabulation Acc, PSW,D**

**The size of the tab is set to**

**Subroutine calling sequence**

**Subroutine name: TABLT**

**Address:**

**Function:**

**Register save:**

**Remarks:**

**eight characters.**

**"TABLT" fills the tabulated part with spaces (20H).**

Subroutine name: TABLAT

|  |  |
| --- | --- |
| Address: | 115AH |
| Function: | Tabulation, but can specify |
|  | the tab code. |
| Input  Register | Contents |
| Acc | Code to tabulate. |
| Register save: | Acc, PSW, C |
| Remarks: | The size of the tab is set to |
|  | eight characters. |

"TABLAT" uses a code specified to Acc for the tabulation. LINE FEED

Moves the screen to the next line. When the cursor is positioned in the bottom line, scrolls one line up within the viewport, but does not change the cursor position.

Subroutine calling sequence

Subroutine name: LFEED

|  |  |
| --- | --- |
| Address: | 10F0H |
| Function: | Line feed. |
| Register save: | Acc, PSW, B,C,D,E,H,L,X,Y |
| HOME POSITION |  |

Moves the cursor to the home position (upper left corner), of the viewport.

Subroutine name: HOMEP

|  |  |
| --- | --- |
| Address: | 13B6H |
| Function: | Home position |

Register save: Acc,PSW, B,C,D,E,H,L,X,Y

CLEAR SCREEN

Fills the screen with null characters (00H). Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine | name: CLRSC |
| Address: | 1393H |
| Function: | Clears the viewport. |

Register save: Acc,PSW,B,C,D,E,H,L,X,Y

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine | name: CLRSCX |
| Address: | 1394H |
| Function: | Fills the viewport. |
|  | with the same code. |
| Input  Registers | Contents |
| Acc | Codes to be established. |

Register save: Acc, PSW,B,C,D,E,H,L

CARRIAGE RETURN

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine name: | CRET |
| Address: | 13CDH |
| Function: | Carriage return |
| Resister save: | Acc, PSW, B,C,D,E,H,L |

"CRET" returns the cursor to the line head. Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine name: | CRETL |
| Address: | 10EDH |
| Function: | Carriage return, line feed |
| Register save: | Acc,PSW,B,C,D,E,H,L |

"CRETL" moves the cursor to the next line head. CANCEL

Fills spaces preceded by the cursor position of the line with null characters.

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine name: | CANCL |
| Address: | 13BBH |
| Function: | Cancel |

Register save: Acc,PSW,B,C,D,E,H,L

2-11 Two modes for one-character display

The one-character display mode has two representations:

Overwrite mode Insertion mode

This manual will refer to the overwrite mode as mode 0 and the insertion mode as mode I.

Specified characters in the mode 0 are displayed at the cursor, on top of characters previously displayed, which are deleted. When characters are displayed in the lower right corner, the characters are scrolled up by one line within the viewport, and the cursor moves to the lower left corner.

When a null character (00H) is displayed in mode I, at the cursor position, the character is displayed in that position, as in mode 0. When a non-null character is displayed, the characters after the cursor in the sentence are shifted backwards by one character, and the new character is displayed in tne cursor position. If the shift results in the deletion of a separation code (null character, 00H) between the shifted sentence and the next sentence, insert one blank line (all null characters) between the shifted sentence and the next sentence. Then, the existing bottom line is deleted. If non-null characters are displayed on the whole bottom line, the characters scroll up by one line within the viewport.

In both modes, if a null character is displayed in the cursor position, all null characters positioned prior to the display in the left side of the viewport are changed to spaces (20H).

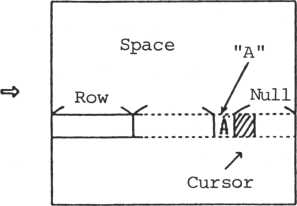
The overwrite mode and insertion mode are selected by DIFLGA DM0DE2 of the processing screen information table.

**If "A" is displayed in the cursor position: Cursor**

**Row**

**Null**

**Null**



Overwrite mode establishment

Subroutine calling sequence

Subroutine name: Address:

STOVRM

**0F19H**

Establishes the overwrite mode.

All registers

Function:

Register save:

Insertion mode establishment

Subroutine calling sequence

Subroutine name 2 STINSM

Address: 0F1EH

Function: Establishes the insertion mode.

Register save: X,Y

A mode once established continues until reestablishment takes place.

2-12 Character read-on screen

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine | name: RDSCH |
| Address: | 14B1H |
| Function: | Gets a character code on |
|  | the screen. |
| Input  Registers | Contents |
| D | Position X |
| E | Position Y |
| Output  Registers | Contents |
| Acc | Character code |

Register save: X,Y

Other conditions: Cy=1 T^e sPecified position

is not in the viewport

This routine gets the character code in a specified position. When the specified position is outside of the viewport, an error occurs.

Subroutine name: RDSCHA

Address: 14ACH

Function: Gets the character code in the

current cursor position.

**Output**

**Registers Contents**

H,L Address in VRAM of the current cursor

Acc Character code

Register save: B,C,D,E,X,Y

This routine gets the character code at the current cursor position.

2-13 One-line display routine

Subroutine calling sequence

Subroutine name: DSPLN

**Address:**

**Function:**

**Input**

**Registers**

B

**H ,L**

**1063H**

**Display**

**Contents Max. number of Top address of**

**one line.**

**characters texts displayed**

**Contents**

**Output**

**Registers**

B

Max. number of characters - number of display characters

Address of last display character + 1 Last display character code C,D,E,X,Y

**H ,L**

**Acc**

**Register save:**

"DSPLN" continues processing until the maximum number of characters set in the B register is output or a terminator (Null or CR) is output. Control code handling depends on the then flag.

Subroutine calling sequence

Subroutine name: DSPLTB

Address: 105BH

Function: One-line display (Continues the

display until a terminator appears)

**Input**

**Registers Contents**

H,L Top address of text displayed

**Output**

**Registers Contents**

H,L Address of last display character +1

Acc Last display character

Register save: X,Y

Remarks: The max. number of display characters is

256 characters. A control code is displayed as a character.

"DSPLTB" limits the number of characters to 256 characters and continues output until a terminator appears within the 256 characters). Control codes are handled as characters.

|  |  |
| --- | --- |
| Subroutine | name s DSPLTA |
| Address: | 105CH |
| Function: | One-line display (displays up  to a terminator) |
| Input  Registers | Contents |
| H ,L | Top address of test displayed |
| Output  Registers | Contents |
| H,L | Address of last display character + 1 |
| Acc | Last display character |

Register save: X,Y

|  |  |
| --- | --- |
| Remarks: | The max. number of display |
|  | characters is 256 characters. |
|  | The control code is executed. |

"DSPLTA" continues output until a terminator appears, and the control code executes the processing.

2-14 One-line read routine

Subroutine calling sequence

Subroutine name: RDSTM

|  |  |
| --- | --- |
| Address: | OEECH |
| Function: | Reads sentences on the screen. |
| Input  Registers | Contents |
| B | Max. number of characters that can |
|  | be input (Data buffer size) |
| D | Input start position X |
| E | H |
| H, L | Input buffer top address |
| Output  Registers | Contents |
| Acc | Last input character (usually return code) |
| B | Max. number of input characters |
|  | - number of characters actually |
|  | input. |
| H,L | (Address of last input characters) + 1 |
| Register save: | C, Y |

Other conditions: C =1 The number of input characters

y

exceeds the data butter size. Reads sentences on the screen.

2-15 Cursor move and cursor out **2-15-1 Cursor move**

Subroutine calling sequence

MV ACS 13DDH

**Subroutine name: Address:**

**Function:**

Changes the cursor position.

**Input**

**Registers**

D

E

**Contents Position X** " Y

**Output**

**Registers Contents**

H ,L Address on the cursor VRAM

Register save: X,Y

Other conditions: -128<x<127 -128<Y<127

Cy=l Other than the viewport were specified.

This routine moves the cursor to a specified position. It outputs the cursor address in VRAM to the H, L registers, and stores it a save area of the cursor address of the processing screen information table.

Subroutine name: STPCU

Address: 1100H

Function: Moves the cursor position

by one character.

**Input**

**Registers Contents**

Acc Move direction n

n = 0 ... Left

1. ... Right
2. ... Up
3. ... Down

Register save: B,C,X,Y

C =1 The cursor is outside Y

**Other conditions:**

the viewport.

The cursor is moved up, down, left, or right in one-character steps to the value set in Acc.

Subroutine calling sequence

|  |  |  |
| --- | --- | --- |
| Subroutine name: | LFTAW, RGTAW, | UPRAW, LWRAW |
| Address: | 10FFH, 10FCH, | 10F9H, 10F6H |
| Move direction: | Left, Right, | Up, Down |
| Register save: | X,Y |  |

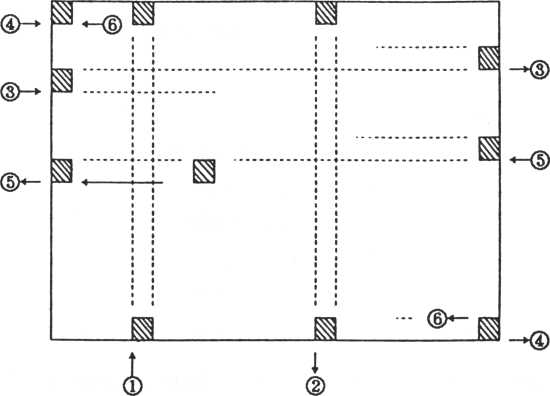
Set a value in Acc to call the preceding "STPCU".

Auxiliary subroutine of cursor move.

"GCURSA" is an internal subroutine of "MVACS" and outputs a cursor address (VRAM) to the HL register.

Moves the cursor up, down, left, or right.

t i



The cursor moves within the viewport as shown above.

1. Move the cursor up. When the cursor moves above the top of the viewport, it reappears at the bottom of the same column.
2. Move the cursor down. When the cursor moves below the bottom of the viewport, it reappears at

the top of the same column.

1. Move the cursor to the right. When the cursor moves past the right end of the viewport, it reappears at the left end one row lower.

(5) Move the cursor to the right. When the cursor

moves past the lower right corner of the viewport, it reappears at the upper left corner.

1. Move the cursor to the left. When the cursor moves past the left end of the viewport, it appears

at the right end one row upper.

1. Move the cursor to the left. When the cursor moves past the upper left corner of the viewport, it reappears at the upper right corner.

The following messages are provided to indicate the next position of the cursor.

Subroutine calling sequence

Subroutine name: STPCUL, STPCUR, STPCUU, STPCUD

Address: 1134H, 1116H, 1147H, 1126H

Move direction: Left, Right, Up, Down

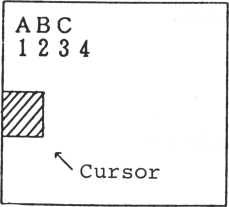
|  |  |
| --- | --- |
| Function: | Calculates the next cursor |
|  | position. |
| Input  Registers | Contents |
| D | Current position X |
| E | " Y |
| Output  Registers | Contents |
| D | Next position X |
| E | " Y |

Register save: B,C,H,L,X,Y

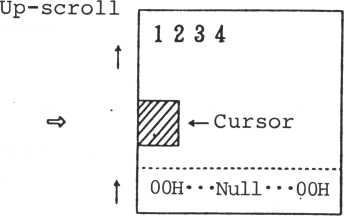
"Cursor out" occurs when the cursor moves out of the viewport. Usually this cannot occur. But, in some cases, the cursor must be moved outside the viewport and must be brought back into the viewport.

2-16 Scrolling and shifting

2-16-1 Scrolling moves rows or columns within the viewport up, down, left, and right. Information deleted from the viewport by the move is not saved. In this case, the cursor position does not change. New rows or columns are filled with null characters.



**Inside the viewport Inside the viewport**



Scrolling up

Subroutine calling sequence

Subroutine name: SCRLF, SCRRG, SCRUP, SCRDW

130AH, 12F9H, 12B3H, 12BFH Left, Right, Up, Down Scrolling

Address:

Move direction: Function

In case of a right scroll, the

left side is filled with spaces (26H)

Y (SCRUP and SCRDW reserve X)

Register save:

The left shift and right shift move a sentence left or right by one character from the cursor position.

**2-16-2 Shifting**

Shifting right

Subroutine calling sequence Subroutine name: SIFTR

|  |  |
| --- | --- |
| Address: | 0FD3H |
| Function: | Right shift |
|  | (Inserts one character in |
|  | the current position and shifts the remaining characters |
|  | rightwards by one character.) |
| Input  Registers | Contents |
| Acc | Character code displayed |
| D | Shift start position X |
| E | " Y |
| Output  Registers | Contents |
| D | Line end position X after shift completion |
| E | ii y " |

Register save: E,X,Y

Other conditions: C =1 Frame over

Acc Last character code at the frame over time

The right shift shifts a character string to the right, from a position indicated by D,E to the line end by one character.

A. A carry is not set.

|  |  |  |
| --- | --- | --- |
| **Acc—** \* A "  **y Shift start \*** **position** |  | **Shift start >/** **position** |
| **~w~** |  | **™ ~~T~** |
| **1 1** |  | **m** |
| / |  | **/** |
| **Line end** |  | **Line end** |
|  |  | **after shift** |

|  |  |  |
| --- | --- | --- |
| **Acc—** **“** B \*  **Shift start y end** |  |  |
| **VA** |  |  |
| **Wa** |  |  |
| **/** |  |  |
| **Line end** |  | **Line end after shift** |

B. A carry is set.

|  |  |  |
| --- | --- | --- |
| Acc— “ C \* |  |  |
| **Shift start y/ position** |  |  |
| **m** |  | c**m** |
| **L** |  |  |
| **/** | |  |

Line end Line end (within viewport)

\* Line end = 00H Left shift

Shifting left

Subroutine calling sequence Subroutine name: SIFTL

Address: OFD3H

Function: Deletes one character and

shifts the remainder of the line left.

**Input**

**Registers Contents**

D Delete position X

E " Y

Register save: X,Y

The left shift shifts character strings left from a position given by D,E to the line end by one character. When there is only a line end, only the line end is shifted left. Even if a specified position is a sentence top, the shift takes place.

Shifting down

|  |  |  |
| --- | --- | --- |
| Shift start **y/** position |  | A character previously positioned in the shift sta^t position.  **\*** |
| **~m** **I** | W |
| m |  | **m** |
| **A**  **/**  Line end | **/**  Line end |

Subroutine calling sequence

Subroutine name: SIFTD

103CH

Address:

Function:

Input Register

E

Register save

Down shift **Contents**

Shift start position X,Y

The down shift moves lines down beyond a line specified by E by one line and fills a specified line with null characters.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |
| II | **Spe- cified line** | **n** |
| II | Null ' |
| **1 1** |  |
|  |  |
|  |

Specified line ~1

2-17 Matrix display and read

2-17-1 Matrix display

Subroutine calling sequence Subroutine name: DSPMX

|  |  |
| --- | --- |
| Address: | 11EDH |
| Function: | Matrix display |
| Input  Registers | Contents |
| B | Number of columns |
| C | Number of rows |
| H, L | Data top address |
| Output  Registers | Contents |
| H,L | Address of last display data + 1 |

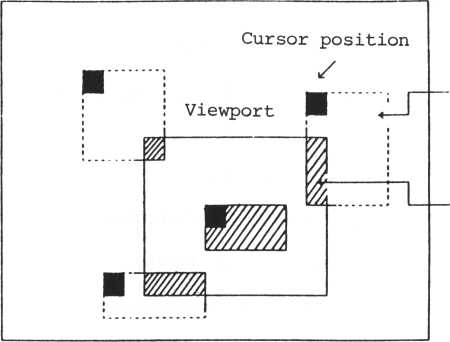
Register save: X,Y

This routine uses a specified column to delimit specified data from the current cursor position and displays the data. The cursor changes data neither before or after the displayed data.

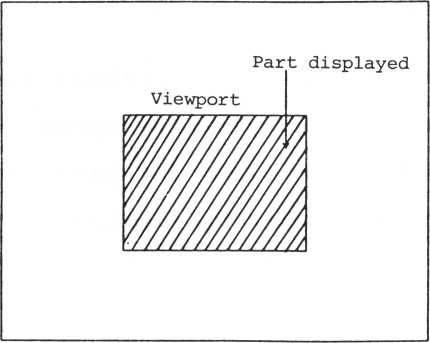
Part outside the viewport is not displayed nor scrolled.

Part not displayed (Outside viewport)

Part displayed



Cursor position



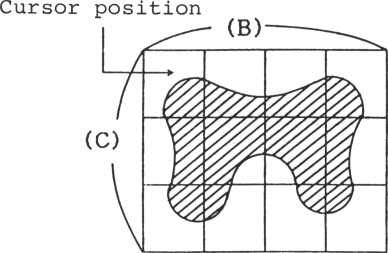
Such an operation procedure as above is also provided.

(H)



**2-17**

**^ Display**



**-2 Row and column read**

**Subroutine calling sequence Subroutine name: RDSMX**

**Address: Function:**

**Input**

**Registers**

**B**

**C**

**H, L**

**Output**

**Registers**

**H, L**

**120EH**

**Reads the matrix.**

**Contents**

**Column**

**Row**

**Top address of read buffer Contents**

**Address of last read data + 1**

**Register save:**

**X,Y**

**This is the reverse procedure to "DSPMX".**

**It reads specified row and column data from the**

**current cursor position. Part run out from the**

**viewport is filled with null character codes.**

**2-18 Sentence definition (as per the screen editor)**

**A sentence is defined as the interval between the left end of the viewport and a null character (00H).**

**2-19 Search of sentence top and next sentence top**

**Subroutine calling sequence**

**Subroutine name:**

**SCTOS**

**Address:**

**119EH**

**Function:**

**Searches the sentence top.**

**Output**

**Registers Contents**

**D**

**Sentence top position X**

**E**

**n**

**y**

**Register save:**

**X,Y**

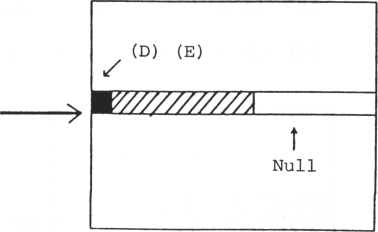
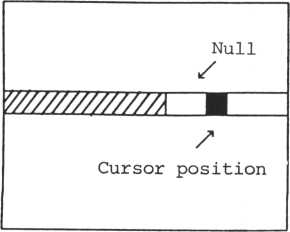
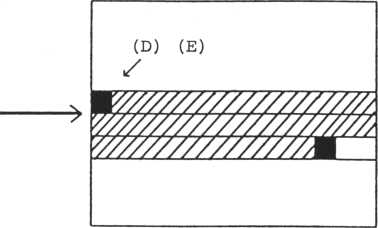
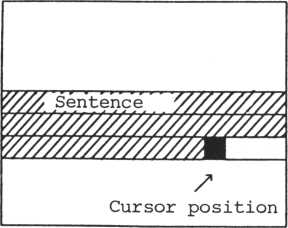


Searches the sentence top and outputs the value to DE. Processing to move the cursor to the top of the sentence will be detailed later ("SCTOSD")

Without a sentence

|  |  |  |
| --- | --- | --- |
| **Sentence Null** 1 / |  |  |
| **W7777777MA** | **'//////////////A** |
| ■ | > | **i** |
| t /  **Null Cursor**  **position** | **A carry is set.** |

Searches for the top of the next sentence.



Subroutine calling sequence

Subroutine name: SNTOS

Address: 11CAH

Function: Searches for the next sentence

top.

**Output**

**Registers**

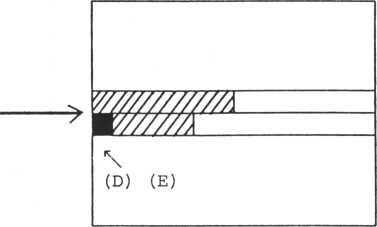
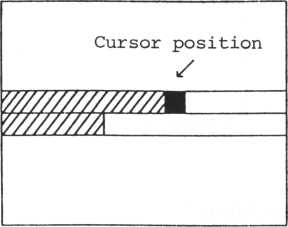
**Contents**

D Next sentence top position X

E " Y

Register save: X,Y

Searches for the top of the next sentence and outputs the location to DE. "SNTOSD" moves the cursor to the top of the next sentence.



|  |  |  |
| --- | --- | --- |
| **Cursor position /** |  |  |
| **y////////////zm** | **W//////////A** |
|  | > | **1** |
| **t**  **Null** | **\ \**  **(D) (E) Nul1** |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **?** |
| **W////M///////////////A** | **W/////////////7777y//////** |
| **W///M** | **I** |

Cursor position

A carry is set.

\

**Subroutine name: Address:**

**Function:**

**Resister save:**

SCTOSD

**13D5H**

**Moves the cursor to the sentence top.**

**X,Y**

**"SCTOSD" moves the cursor to the top of the sentence. This corresponds to control code "02H" (control "B").**

**Surboutine calling sequence**

**Subroutine name: SNTOSD**

**Address: 13DAH**

**Function:**

**Moves the cursor to the top of the next sentence.**

Register save:

**X,Y**

**"SNTOSD" moves the cursor to the top of the next sentence. This corresponds to control code "OEH" (control "N").**

This section describes transfer instructions between the main memory and VRAM.

**2-20 Transfer instructions**

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine | name: PBVRAM |
| Address: | 14BDH |
| Function: | Writes 1 byte of data to VRAM |
| Input  Registers | Contents |
| Acc | Data written |
| H,L | Address written |

Register save: All registers

This subroutine writes one byte into VRAM.

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine | name: GBVRAM |
| Address: | 14C5H |
| Function: | Reads 1 byte data from VRAM. |
| Input  Registers | Contents |
| H,L | Read-out address |
| Output  Register | Contents |
| Acc | Read-out data |

Register save: Others than Acc

This subroutine reads one byte from VRAM, PBV, and GBV.

Subroutine name: PBVRID

|  |  |
| --- | --- |
| Address: | 0010H |
| Function: | Output 1 byte to VRAM |
| Input  Registers | Contents |
| Acc | Output data |
| H,L | VRAM address |

Register save: All registers

|  |  |
| --- | --- |
| Remarks: | Using RST2 (Code D7H) |
|  | The difference infunction of |
|  | PBVRAM is that there is a no |
|  | interrupt condition. |

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine | name: GBVRID |
| Address: | 0018H |
| Function: | Read-out 1 byte data from VRAM |
| Input  Registers | Contents |
| H,L | Read-out address |
| Output  Registers | Contents |
| Acc | Data |

Register save: Others than Acc

|  |  |
| --- | --- |
| Remarks: | Using RST3 (Code DFH)  There is a no-interrupt con |
|  | dition . |

. Block transfer instruction

The block transfer instruction consists of the follow' ing:

Main memory VRAM

VRAM Main memory

VRAM VRAM

Depending on the operation, others are available.

1. Main memory

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine | name: WDVPM |
| Address: | 14 6 OH |
| Function: | Transfers data from main memory to VRAM. |
| Input  Registers | Contents |
| H ,L | Data buffer top address (CPU memory) |
| D,E | VRAM destination address |
| B | Number of bytes of transfer data |

Register save: X,Y

"WDVPM" writes data from main memory to VRAM. The counter is one byte.

|  |  |
| --- | --- |
| Subroutine | name: CVTIR |
| Address: | 0E61H |
| Function: | Transfers data to VRAM. |
| Input  Registers | Contents |
| H,L | Data buffer top address |
|  | (CPU memory) |
| D,E | Destination address |
|  | (VRAM) |
| B ,C | Number of bytes of transfer data |

Register save: X,Y

"CVTIR" outputs data from main memory to VRAM. The counter is two bytes.

1. VRAM Main memory

Subroutine calling sequence

Subroutine name: RDVPM

|  |  |
| --- | --- |
| Address: | 144BH |
| Function: | Reads out VRAM data. |
| Input  Registers | Contents |
| H,L | Top address of read-out VRAM data |
| D,E | Top address of CPU data buffer |
| B | Number of bytes of read-out data |

Register save: B,C,X,Y

"RDVPM" transfers data from VRAM to the main memory The counter is one byte.

Subroutine name: VCTIR

|  |  |
| --- | --- |
| Address: | 0E7DH |
| Function: | Reads data from VRAM. |
| Input  Registers | Contents |
| H, L | Top address of read-out VRAM data |
| D, E | Top address of CPU memory data buffer |
| D, C | Number of bytes transferred |

Register save: X,Y

"VCTIR" transfers data from VRAM to the main memory The counter is two bytes.

1. VRAM -\* VARM

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine | name: BLKMV |
| Address: | 0B75H |
| Function: | Transfers data between VRAM |
|  | addresses. |
| Input  Registers | Contents |
| H,L | Source address in VRAM |
| D, E | Destination address in VRAM |
| B, C | Number of bytes transferred |

Register save: X,Y

"BLKMV" is a routine to transfer data between VRAM locations. It uses a two-byte counter.

Subroutine name: BLKMV2

|  |  |
| --- | --- |
| Address: | 0B81H |
| Function: | Transfers data between VRAM |
|  | locations. |
| Input  Registers | Contents |
| H,L | Source address in VRAM |
| D**,** E | Destination address in VRAM |

Register save: X,Y

|  |  |
| --- | --- |
| Remarks: | Number of transferred bytes: |
|  | 2 KB (fixed) |

"BLKMV2" operates on 2 KB of data at a time and transfers VRAM tables.

1. Others

Subroutine calling sequence

Subroutine name: PADVRM

|  |  |
| --- | --- |
| Address: | 0E01H |
| Function: | Outputs the same code con |
|  | tinuously to VRAM. |
| Input  Registers | Contents |
| Acc | Output code |
| BC | Number of bytes output |
| H**,** L | Destination top address in VRAM |

Register save: E,X,Y

"PADVRAM" fills an area specified by VRAM with the same code.

This system provides two multiplication routines, but does not consider overflow, etc.

**2-21 Multiplication routine**

Subroutine calling sequence

Subroutine name: MLTAL

|  |  |
| --- | --- |
| Address: | 1441H |
| Function: | L register x Acc register |
|  | (unsigned multiplication) |
| Input  Registers | Contents |
| L | n (0 ^ n ^ 255) |
| Acc | m (0 ^ n ^ 255) |
| Output  Registers | Contents |
| H, L | n x m |

Register save: B,C,D,E,X,Y

Subroutine calling sequence

Subroutine name: MULTHD

|  |  |
| --- | --- |
| Address: | 142CH |
| Function: | HL register x DE register (unsigned multiplication) |
| Input  Registers | Contents |
| H, L | n (0 ^ n < 65535) |
| D, E | m (0 £ n 65535) |
| Output  Registers | Contents |
| H, L | n x m |

Register save: B,C,X,Y

2-22 Other information

Routines not described in previous sections are detailed below.

Subroutine calling sequence

Subroutine name: Address:

Function: Register save:

VDPINT

OEOBH

Initializes VDP. X,Y

Sets both page 0 and page 1 or VRAM to the GI mode for initialization.

Subroutine calling sequence

Subroutine name: EXTBL

1076H

**Address: Function:**

**Input**

**Registers**

**Acc**

**H ,L**

**Register save: Remarks:**

**Jump table top address (H,L)**

Table jump

**Contents**

Table offset

Jump table top address

: B,C,E,X,Y

Jump table

**This routine executes a table**

|  |  |
| --- | --- |
| 1st processing routine | L |
| top address | H |
| 2nd processing routine | L |
| top address | H |
|  | L |
|  | H |
| nth processing routine | L |
| top address | H |
|  | L |
|  | H |

When jumping to the nth processing, set n to Acc

jump,

The following processing determines whether a cursor out has occurred.

Subroutine calling sequence

Subroutine name: CCUROT

|  |  |
| --- | --- |
| Address: | 1481H |
| Function: | Checks for a cursor out. |
| Input  Registers | Contents |
| D | Position X |
| E | Position Y |

Other conditions: Cy=l Cursor out

(outside of viewpoint)

Chapter 3 Keyboard Handling

CONTENTS

1. [Ten-shift key matrix 3-1](#bookmark43)
2. Auto-repeat function 3-10
3. [Cursor and cursor blinking 3-10](#bookmark48)
4. Reset/halt keys 3-11
5. [Type ahead mode 3-12](#bookmark49)
6. [Entry (audio) confirmation 3-13](#bookmark50)
7. [Keyboard time-out 3-13](#bookmark51)
8. [Function key setting 3-13](#bookmark52)
9. [Joypad and attack switch 3-16](#bookmark53)
10. 1 Ten-shift key matrix

This computer has a full keyboard with 54 keys. The keyboard can enter more than 54 characters, however, by using multifunction keys. Characters are classified into several groups, each group of keys sharing a general function. In this manual, this general function will be called a "mode."

There are three basic modes:

1. Letter mode (with lowercase characters

the standard entry)

1. Capital mode (with uppercase characters

the standard entry)

1. Graphics mode (input pseudo-graphics)

Modes are set to make a key perform a different function. The "s" key is pressed, for example, to enter both a lowercase "s" and an uppercase "S." The difference is, of course, that the shift key is also pressed. The shift key is one of the "auxiliary" keys that are used to change the function of a key. There are three auxiliary keys:

1. Shift key (SHIFT)

Two keys: one on the right and one on the left

1. Control key (CTRL)

One key

1. Function key (FUNC)

One key

The characters and keys are in a one-to-one correspondence in any given mode.

The auxiliary keys have no meaning when they are used alone. They are meaningful only when they are pressed at the same time as a character key.

The shift keys assign the alternate characters allocated to the keys in each mode. Control codes assigned to keys do not change with the mode.

When a character control code is entered by pressing a key in the letter mode and the control key, the D6 bit is set to 0 (zero). But, some of the keys (such as the cursor move key) are handled as exceptions.

Character-strings, not characters, are entered using the function key. The character strings can be defined freely by the user. The character strings can be used for words reserved in the language utility and for the user optional definition key. A maximum of 26 character strings is available. One function can be set to equal a 255- character string.

This computer interprets the keyboard shift in combination with the following modes and auxiliary keys.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1. | Letter | mode | Numbers | and | lowercase | letters |
| 2. | Letter | shift | mode Symbols | and | uppercase | letters |
| 3. | Capital | . mode | Numbers | and | uppercase | letters |

Pseudo-graphics (80H to 9FH)

Pseudo-graphics (EOH to FFH)

1. **Graphics mode**
2. **Graphics shift mode**
3. **Control**
4. **Control shift**
5. **Function**

**10. Function shift**

Enters a control code

Handles a control code as a character

Displays and executes a pre-set character string

If there is a control code

in the saved character string,

does not execute the code but

displays it as a character.

Each key has one number, which is called the key address. The address is expressed as follows.

Address = |(input port) AND FH | \* 8 + (the bit  
allocated to the key) +1

Addresses allocated to each key, keyboard matrixes and the correspondence between matrixes and characters in each mode are listed below. For the keyboard arrangement and standard graphics fonts, see the Attachement.

Correspondence between keys and addresses

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 41 | 42 | 43 | 44 | 48 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 49 | 50 | 51 | 52 | 8 |
|  | | | | | | | | | | | | | |
| 0 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 53 | 54 | 55 | 56 | 7 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 45 | 46 | 47 | 0 |

Address 0 means no input.

Address

**30H 31H 32H 33H 34H 35H 36H 3 7H**

|  |  |
| --- | --- |
| D7 | |
|  | 8  1 |
|  | 1  16 |
|  | 24 |
|  | 32 |

**7-0**

15

23 -

- 31

**0-0**

**- 0**

14

- 13

12 - 11

10

22 - 21 - 20

- 19

18

- 17

- 30

29 - 28

27

26-25

I

40 - 39 - 38

37

36 -

35-34

- 33

48

47

46 - 45

44

43 -

42 - 41

56 — 55 — 54 — 53 — 52 —

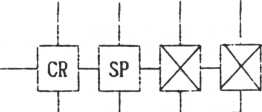
51-50

IZJZZL

49

shiftMSift

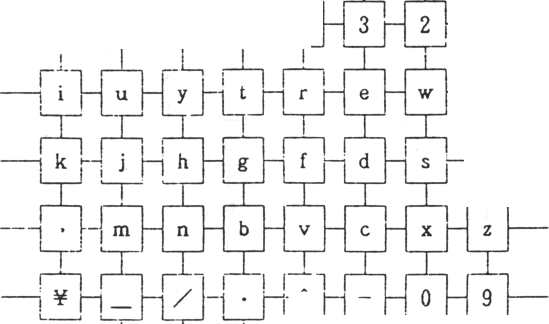
right1 left



|  |  |  |
| --- | --- | --- |
|  |  | |
| CTRL | |  |
|  |
|  | | |
| 1 | |  |
|  |
|  | | |
| g | |  |
|  |
|  | | |
| a | |  |
|  |

FUNC

-i\*ivMTl-m-u



36H

37H

Uppercase letters in the capital mode,

' XO ' XI ' X2 ' X3 ' X4 ' X5 ' X6 ' X7

Lowercase letters in the capital mode.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CR -**  nr | **-4** |  | **-x-** | **\_SHIFT\_**  **right** | **SHIFT \_ left**  **|** | **- FUNC -**  **|** | **- CTRL**  rr |
| **(-** | **...** .1 **,** v | **L—**  - &L -  L~rJ | J r | **- $ !p** | **- # -**  **nr** | **——L- »**  1 | **!** |
| **i -**  1 | 1  **- u -** | **i**  **- Y -** | **- T -** | **- R -** | 1  **- E -** | **...** 1**.**  **- W -** 1 | **- Q**  **ltj** |
| 1  **K -**  1 | **"Lj-T**  1 | 1  iH r | **- G -** 1 | **r** | **- D -** 1 | **L**  **- s -**  **nr** | 1  **- A**  1 |
| **.,1**  **< "** | **L\_**  **- M -**  **n^** | **- N - n-"** | **—-L—**  **- B -** | ■f | .1  **- C -** | **- x -**  v | **...** .1 **.. - z** 1 |
| **1**  **1** | **-L.** | **..L**  **- ? -** | **.1.**  **- > -** | **— — n^** | **nr** | **1**  **X,**  **1** | **1**  **-**) |
| } **- LT“J** | **L**  **- \* -** | **- -b -** | **.1**  **- L -**  V | **I**  - {**-** | **1**  **Sr** | **J-**  **- P -** | **1**  - 0 V |
| **!** |  |  | **1** | **-** Yj- | **i** |  | **1** |

**30H**

**31H**

**32H**

**33H**

**34H**

**35H**

**36H**

**37H**

CR

- SP -

-m-

93 -1 92

97 - 96

9B

-91-90

- 95 -

9fl

84 - 81

9E

x

**r9F**

|SHIFT|\_

right

**94j—J^88j-**

99 - 98

**h\_**

80

**Vf**

j L.

9D

r

|  |  |
| --- | --- |
| — FUNC -  J | - CTRL  L~rJ |
| L  - 2 - I | 1,  -1  nr |
| , l .  - 86 - J | -l— - 8C |
| 1.  - 8B -  I | 1  - 8fl  L~rJ |
| . 1 \_  -j 85 -  1 | . l..... - 8F |
| 1  - 0 -  ' | Lf |
| . L\_  -j 83 -  -i | -1 82 |
| 5 |

SHIFT!

left

87 - 8D

8E

- 9C

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **30H** | **CR - 1** | **- SP -** |  | **-x-** | **Jshift!**  **jrigiit]** | **SHIFT \_**  **left**  **1** | **- FUNC - 1** | **- CTEL** |
| **31H** | **. -L ( -** |  | **- & -**  {~TJ | **-** % **- v** | **- $ -** l~TJ | **-** it **-** L~rJ | **1.**  ■n | **- !** |
| **32H** | **1**  **F3 -** | **1.**  **- F2 -** | **.■JL. - FI -** | **. L**  **- FO -** | **1**  **- E7 -**  **ltj** | **i**  **- ED -** | **\_i**  **- E6 - 1** | **- EC**  **nr** |
| **33H** | **1**  **F7 -**  **hr** | **rh**  **t!ir** | **- F5 -**  **1** | **F4 -** | **1**  **- E8 -**  **hr** | **1**  **- E9 - |** | **1.**  **- EB -**  **hr** | **1**  **- EA**  **1** |
| **34H** | **< -** | **- FB -**  **hr** | **- FA -**  **V** | **- F9 -** | **- F8 -**  **hr** | **- EE -**  **hh** | **- E5 j- 1** | **I**  **- EF** |
| **35H** | **i**  **i**  L~rJ | **- E4 -** | **1**  **- El -** | **- > -**  {~TJ |  | **1** | **1** | **- ) S-** |
| **36H** | **i**  **FE - 1** | **-j FF -** | **JL.** | **1**  **- EO -**  **—**r—\* | **- FD -** | **1**  **- FC - 1** | **J.**  **- E3 -** | **■1- - E2** |
| **37H** |  | **5-** | **-y-** | **-5-** | **-y-** | **-y-** | **-5-** |  |

The auto-repeat function allows repeated input of the same character simply by pressing the key longer than normally.

There are two auto-repeat counters: one for the autorepeat start time and one for the auto-repeat interval. The auto-repeat start time is counted after the start of key input until auto-repeat starts. The auto-repeat interval is counted from the auto-repeat start.

The default value is 0.5 seconds for the auto-repeat monitor time and 66 milliseconds for the auto-repeat interval.

1. 3 Cursor and cursor blinking

3-3-1 Cursor pattern

The cursor is displayed at the position corresponding to the cursor position in the screen information table, and it blinks at a fixed rate.

Patterns displayed on the cursor position are determined as listed below for each mode.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Letter mode** | | | | **Capital mode** | | | **Graphics mode** | | |
| **Over-writing**  **mode** |  | | | |  | | |  | | |
|  |  | **“i** |  |  | **©** |  |  | **©** |  |
|  | | | |  | | |  | | |
| **Insertion**  **mode** |  | | | |  | | |  | | |
|  |  |  |  |  | **©** |  |  | **gj** |  |
|  | | | |  | | |  | | |

The RESET key is effective only when the shift key is pressed. The HALT key is effective only when the control key is pressed.

Users can handle the processing as desired when the RESET and HALT keys are pressed. For this purpose, a head address must be allocated for system processing. Usually pressing these keys causes the system to return to the command input status.

Stack status at RESET and HALT key interrupt.

When the RESET or HALT key is pressed, and control returns to the system, the stack is set as below.

Hence, the user must operate to adjust the stack level.

When control passes to the RESET and HALT key processing, data saved in each register becomes meaningless.

1. 5 Type ahead mode

|  |  |
| --- | --- |
| **Register** | **L** |
| **ii** | **H** |
| **Register** | **E** |
| **H** | **D** |
| **Register** | **C** |
| **H** | **B** |
| **PSW** |  |
| **(Acc)** |  |

**The return address used when the RESET or HALT key is pressed.**

**The register group reserved at a CTC interrupt.**

In the type ahead mode, characters in the keyboard buffer are input in parallel with the instructions generated by the interrupt processing except when accepting input from the keyboard.

The keyboard buffer size is 64 bytes in a ring buffer format.

The KINFLG bit 0 must be changed when the user sets or releases the type adhead mode.

3-6 Entry (audio) confirmation

Key entry is confirmed by an audible click sounding at the same time as the key input.

The click frequency and length can be changed by the user. Turning the click switch (KINFLG bit 7) OFF can stop the click.

3-7 Keyboard time-out

If no input is made within a set wait time after the previous input, a time-out error occurs. The key input time is set by the key input wait in the keyboard information table and standard key input time.

3-8 Function key setting

The procedure for setting the function key is as follows.

1. Set a function key control flag of the function key control table in the system table.

**76543210**

Number of function keys (max. of 26)

Function key data format

The data format is:

**0**:

**Counter**

**Character string**

Put the counter in the first byte of the character string. Then, the number of character strings indicated by the counter follows.

Is

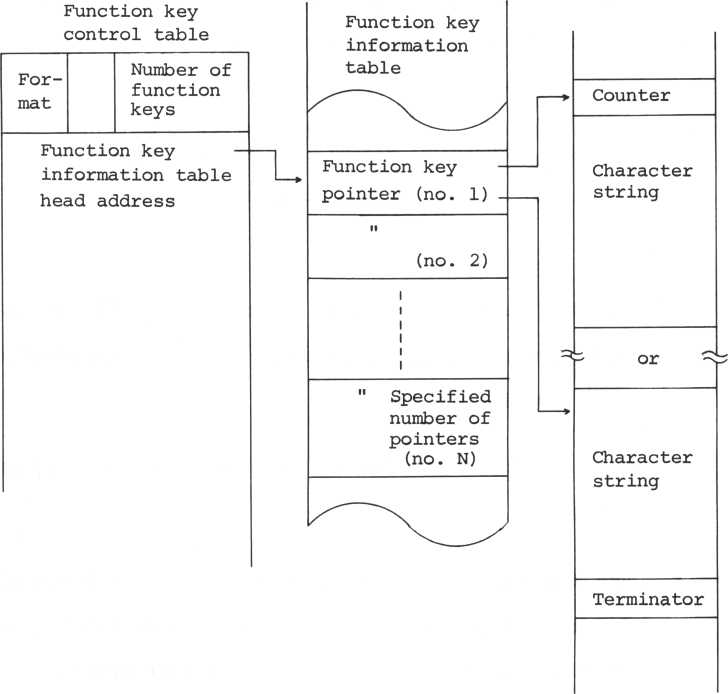
**Character string**

**Termi**

**nator**

Put the terminator (null character or CR) at the end of the character string. But the terminator does not ignore format 0.

1. Save the head address of the function key information table.
2. The function key information table uses a saved function key x 2 byte table for specified character string pointers in increments of two bytes.
3. Save data in the format specified by the function key control flag at the position specified by the pointer.



3-9 Joypad and attack switches

The joypad has four direction switches and two attack switches.

The four direction switches provide data for any of eight directions or a neutral state.

There are two attack switches, which can be expanded to four switches.

3-9-1 Joypad support

If system interrupt support is desired for the joypad direction switches, set the joypad switch of the event control flag to 1. If this switch is set to 0, system support cannot be provided, but if the user calls the subroutine (JOYSP) for joypad support, the joypad direction number can be read.

3-9-2 Attack switches

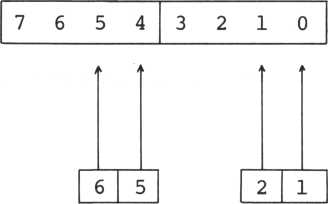
Since the attack switch is not supported by the system, the user must read data directly from the I/O port.

Port name: KEYMD1

Attack switch data:

Out of the 8 data bits, bits 0 and 1 are used for the left attack switch and bits 4 and 5 for the right attack switch.

**MSB LSB**



V v / \ v )

**Right Left**

Bits 2, **3,** 6, and 7 are not used.

Note: The attack switch utilizes the keyboard numerics

for the same addresses as the keys, and the attack switch is read during the keyboard scanning. Hence, pressing the attack switch during the keyboard scanning causes numerics to be input.

**Subroutine name**

**WTKDTC**

**Address:**

**0756H**

**Function:**

**Blinks the cursor and waits**

**for key input.**

**Output**

**Registers Contents**

**Acc**

**Input character code**

B

**Auxiliary key information**

**Register save**

**D,E,X,Y**

**Other conditions: Cy=l Time-out error**

**This subroutine blinks the cursor and waits for data input into the keyboard buffer. When the time-out flag (down-counter end flag) becomes 1 before the data is input, it is regarded as a time-out and is handled as a time-out error. When an error occurs, stop the down-counter and delete the flag (reset it to 0).**

**Subroutine name**

**SCNKB**

**Address**

**0966H**

**Function:**

**Keyboard scan**

**Output**

**Registers Contents**

**Acc**

**Pressed key address**

**B**

**Auxiliary key information**

**Register save:**

**D,E,H,L,X,Y**

**Other conditions: Cy=l A key being pressed**

**is not found.**

**This subroutine scans the keyboard and provides the address of the key being pressed. Auxiliary key information is output to register B. If a key is not pressed, a carrier is set and the key input is not waited for.**

Subroutine name: ACECH

Address: 0827H

Function: One-character input

**Contents**

**Input**

**Registers**

**Acc**

**Output**

**Registers**

**Acc**

**B**

Echo back flag 0 : without echo-back 1: with echo-back

**Contents**

Input character code Auxiliary key information

Register save: E,X,Y

Other conditions: Cy=l Time-out error

One-character input processing from the keyboard.

If the input is not provided within the time period determined by the keyboard input time of the keyboard information table, a time-out error occurs.

When an echo back flag is set, the interrupt characters are displayed on the screen.

When this processing is accessed, first the keyboard timer starts. If already during the operation, the counter under operation continues.

One-character input

**Function:**

**Output**

**Registers Contents**

Input character code

**Acc**

**B**

Auxiliary key information

Register save:

**D,E,X,Y**

**Subroutine name:**

**ACECHI**

**Address:**

**0845H**

Other conditions: Cy=l Time-out error

This subroutine gets one character from the keyboard. Differences from ACECH are listed below.

. The keyboard timer is not started. (keeps the current status)

. Echo-back is not used.

. Only registers D, E and X, Y are saved.

Subroutine names EDTLN

|  |  |
| --- | --- |
| Address: | 0668H |
| Function: | Edit input |
| Input  Registers | Contents |
| B | Max. number of input characters |
|  | (Input buffer size) |
| H ,L | Input buffer head address |
| Output  Registers | Contents |
| Acc | Last input character code (Usually a terminator) |
| B | Max. number of input characters |
| H,L | Next address of the last input |
|  | character |

Register save: X,Y

Other conditions: Cy=l Time-out error

|  |  |
| --- | --- |
|  | (An error code is input |
|  | into Acc.) |
|  | Input data exceeded |
|  | the buffer size. |

This subroutine edits and inputs statements displayed on the screen.

Subroutine name: EDTST

|  |  |
| --- | --- |
| Address: | 0689H |
| Function: | Edit and input |
| Output  Registers | Contents |
| D | Head position X of inputted statements |
| E | »i **y "** |
| Register save: | : X,Y |

Other conditions: Cy=l Time-out error

This is an internal subroutine of EDTLN, and it processes the portion between the edit start and the return input.

Subroutine name: ACEST

|  |  |
| --- | --- |
| Address: | 066FH |
| Function: | Edit and input |
| Input  Registers | Contents |
| B | Max. number of input characters |
|  | (input buffer size) |
| D | Input start position X |
| E | " Y |
| H, L | Input buffer head address |
| Output  Registers | Contents |
| Acc | Last input character code |
| B | Input buffer size |
|  | - Number of input characters |
| H ,L | Last input character address + 1 |

Register save: X,Y

Other conditions: Cy=l The input data exceeded

the buffer size.

This is an internal subroutine of EDTLN, and it processes the portion to read statements input after the return.

EDTLN is processing in a combination of EDTST and ACEST. It is possible for the utility to use these two kinds of processing to arrange EDTLN.

Subroutine name: CALKAD

|  |  |
| --- | --- |
| Address: | 097BH |
| Function: | Key address computation |
| Input  Registers | Contents |
| Acc | Input key data |
| C | I/O address |
| Output  Registers | Contents |
| Acc | Pressed key address |

Register save: B,C,D,E,H,L,X,Y

Other conditions: Cy=l When the key is not

pressed.

This subroutine computes the key address for a pressed key. The address expression is as follows.

Address = | (I/O port No.) AND 7 | \* 8 + (the bit

allocated to the key being pressed) + 1

Subroutine name: DECTR

|  |  |
| --- | --- |
| Address: | 090BH |
| Function: | Decode in control mode |
| Input  Registers | Contents |
| Acc | Key address |
| B | Auxiliary key information |
| Output  Registers | Contents |
| Acc | Decoding result |
| B | Auxiliary key information |

Register save: B,C,X,Y

Other conditions: Cy=l No key is allocated.

This subroutine gets characters allocated to keys in the control mode.

Subroutine name: DECFN

|  |  |
| --- | --- |
| Address: | 0933H |
| Function: | Function key decoding |
| Input  Registers | Contents |
| Acc | Key address |
| B | Auxiliary key information |
| Output  Registers | Contents |
| B | Auxiliary key information |
| H,L | Head address of a character string |
|  | allocated to the function key. |

Register save: X,Y

Other conditions: Cy=l No key is allocated.

This subroutine gets the head address of a character string allocated to a key in the function mode.

Subroutine name: DECAD

|  |  |
| --- | --- |
| Address: | 08DAH |
| Function: | Address decode |
| Input  Registers | Contents |
| Acc | Key address |
| B | Auxiliary key information |
| Output  Registers | Contents |
| Acc | Decoded result (pressed character) |
| H,L | In case of the function shift, |
|  | this indicates a head address of |
|  | an allocated character string. |

Register save: B,C,D,E,X,Y

Other conditions: Cy=l No character is allocated

This subroutine gets the characters allocated to a pressed key from the key address and auxiliary key information.

Subroutine name: CMPCUR

Address: 0784H

Function: Comparison of two coordinates

**Input**

**Registers**

**Contents Coordinate 1** " **2**

H,L D, E

Register save: B,C,D,E,H,L,X,Y

Remarks: Cy=l Z = 0: Coordinate 1 < Cordina-

Cy=0 Z=1:

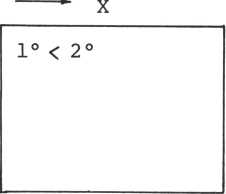
Cy=0 Z-l: " >

This subroutine interprets the contents of register pair (H,L) and the most-significant bytes of (D,E) as X and the least-significant bytes as Y, and compares them.

If the Y coordinates of coordinates 1 and 2 coincide, tl coordinate size depends on the coordinate X size.

Unless the Y coordinates of coordinates 1 and 2 coincide the coordinate size depends on the coordinate Y size without regard to coordinate X.

**(Ex)**



X

**1**°

2°

**Y**

1> 2

|  |  |
| --- | --- |
| Subroutine | name: STRTKT |
| Address: | 0992H |
| Function: | Keyboard timer start. |

Register save: B,C,D,E,H,L,X,Y

Other conditions: Cy=l Already started.

This subroutine starts the timer at keyboard input The key input wait time and the standard time are used to determine the time-out.

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine | name: ACELN |
| Address: | 07A9H |
| Function: | One-line input |
|  | (without edit function) |
| Input  Registers | Contents |
| B | Input buffer size |
| H,L | Input buffer head address |
| Output  Registers | Contents |
| Acc | Last-input character |
| B | Max. number of input characters |
|  | - Number of input characters |
| C | Number of input characters |
| H ,L | Last-input character address + 1 |

Register save: X,Y

Other conditions: Cy=l Time-out error

The input data exceeded the input buffer size.

This subroutine provides line input processing (editing) with one-character deletion and line deletion only. The following characters can be used as terminators.

. Return (ODH)

. Return (17H)

. Counter control key (1CH 1FH)

. Home (OBH)

. Escape (1BH)

. Line feed (OAH)

The terminator is placed in Acc at the processing end time and is simultaneously saved in TERMAL of the key input information table also. Terminator echo-back takes place only for the return.

This processing is most efficient, not in getting input information from the screen, but in accepting only information entered directly from the keyboard.

|  |  |
| --- | --- |
| Subroutine | name: PAD |
| Address: | 0861H |
| Function: | Padding |
| Input  Registers | Contents |
| Acc | Padded data |
| H**,** L | Padding start address |
| B,C | Number of bytes padded |
| Output  Registers | Contents |
| H, L | Last padded address +1 |

Register save: Acc,X,Y

This subroutine pads the specified memory space with the codes specified in Acc.

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine | name: NULPAD |
| Address: | 086 OH |
| Function: | Null clear |
| Input  Registers | Contents |
| H,L | Clear start address |
| B,C | Number of bytes cleared |
| Output  Registers | Contents |
| H,L | Clear end address + 1 |

Register save: X,Y

This subroutine fills the main memory with null characters

Subroutine name: GTKDT

Address: 08ACH

Function: Gets one character from the

keyboard buffer.

**Output**

**Registers Contents**

Acc Non-numeric data

Auxiliary key information (control code)

**B**

Register save: X,Y

Other conditions: Cy=l No data exists.

This subroutine gets one character of information from the keyboard buffer. In case of characters other than the control code, 00H is put on the auxiliary key information.

Cancells data saved in the keyboard buffer.

**Function:**

**Register save**

**Subroutine name**

**CLKBF**

**Address**

**077BH**

PSW,B,C,D,E,H,L,X,Y

This subroutine effectively "sets" all information in the keyboard buffer to null characters. Actually, the data is not erased; a get pointer value is assigned to the put pointer. This does, however, have the same effect and the range of effective data becomes unknown. When the keyboard "pre-pressing" assignment is released, this subroutine must be executed before any information from the keyboard buffer can be accepted. Hence, the function key handles only one head character, and the effectiveness is lost.

Subroutine name: CHKYM

|  |  |
| --- | --- |
| Address: | 073FH |
| Function: | Checks the key input mode |
|  | change. |
| Input  Registers | Contents |
| Acc | Control code |
|  | 01: Letter mode |
|  | 02: Capital mode |
|  | 03: Graphics mode |
| B | Auxiliary key information |

Register save: D,E,X,Y

Other conditions: Cy=l This is not a key

input change.

This subroutine uses the auxiliary key information and input character data to detect a change in key input

|  |  |
| --- | --- |
| mode assignment, changes mode. | Upon detection, the subroutine |

Changing the input mode requires the MSB of the auxiliary key to be 1.

Subroutine name: STDMl

|  |  |
| --- | --- |
| Address: | 04FFH |
| Function: | Sets display mode 1 |
| Input  Registers | Contents |
| B | Auxiliary key information |
| Remarks: | Z = 0 DMDL = 1 |
|  | Z = 1 DMDl = 0 |

The display mode 1 is a flag to select execution of the control code (flag is 0) or the display of it (flag is 1). Refer to the section of the Display Handler for DSPCH.

Chapter 4 Sprite Handler

CONTENTS

1. [Sprite pattern setting 4-1](#bookmark55)
2. [Resolution and magnification of sprites 4-4](#bookmark56)
3. [Contents of the sprite attribute table 4-7](#bookmark57)
4. Sprite position (physical position

and logical position) 4-10

1. Sprite move 4-15
2. Sprite deletion 4-18
3. Sprite information from VDP 4-20

Sprite pattern setting

1 Sprite characteristics

**4-1**

**4-1-**

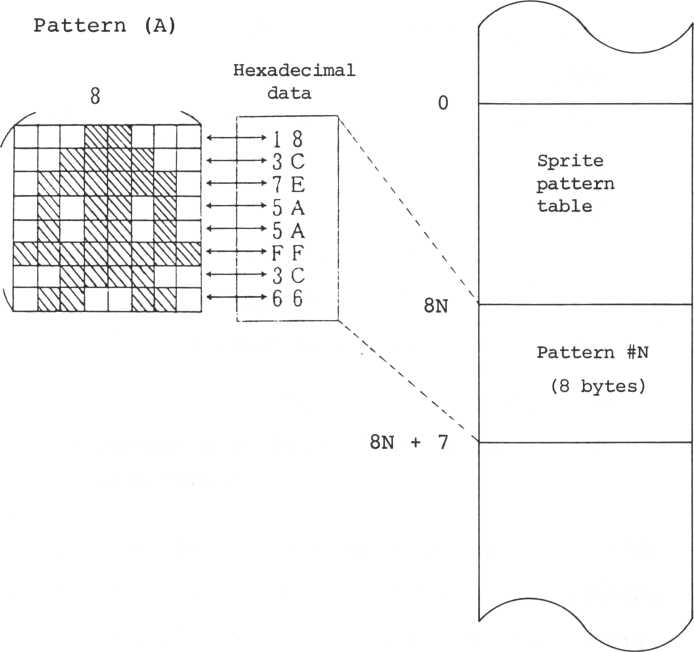
The sprite is an graphics pattern that is hardware-

supported by VDP.

A sprite has three basic characteristics:

1. It can freely set coordinates X and Y without regard to the character plane.
2. It has a priority hierarchy on display of 32 sprites. When a lower priority sprite contends with a higher priority sprite at the same location the lower priority sprite is deleted automatically
3. It enables the three-dimensional graphics to be generated easily.

Thus, the sprite feature is very convenient. To display a sprite, the sprite pattern must be set in VRAM.



When the above sprite pattern (A) is set as the sprit pattern #N**,** write data into the position indicated by sprite pattern table #N in VRAM. This is the same procedure as for the character set. Up to 256 def- ferent sprite patterns can be set (#0 to #255) .

Subroutine name: STSCHR

|  |  |
| --- | --- |
| Address: | OE58H |
| Function: | Sets the sprite pattern. |
| Input  Registers | Contents |
| B | Number of patterns to be set. |
| C | First set code no. |
| H,L | Data head address |

Register save: X,Y

Other conditions: Cy=l The parameter is

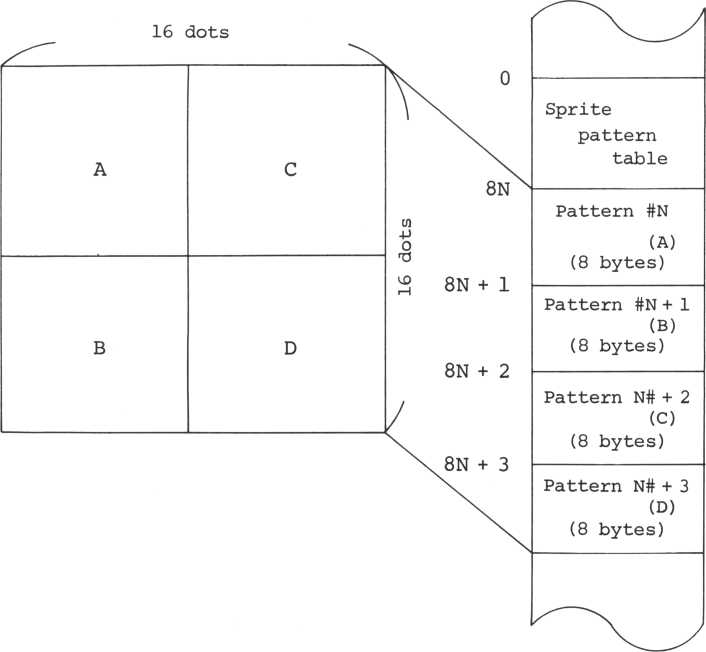
inadequate.

This subroutine sets the sprite data from the sprite pattern table head in increments of eight bytes by sprite code numbers set in the C registers during this processing.

1. 2 Resolution and magnification of sprites

Sprites come in two sizes: vertical 8 dots x horizontal 8 dots and vertical 16 dots x horizontal 16 dots.

Definition of the 8 x 8-dot sprite was detailed in Section 4-1. This section describes how to define a 16 x 16-dot sprite.



The 16 x 16-dot sprite requires 32 bytes for one pattern. When setting the 32 bytes, divide the pattern into four parts and set the two left blocks then the two right blocks as shown above. Thus, if this procedure is

applied to STSCHR on the preceding page, four smaller patterns must be defined to set one complete pattern.

In this case, it should be noted that giving any code of N, (N + 1), (N + 2) , and (N + 3) to the attribute table causes the same display as given N.

A sprite can be of either of two sizes: one-time magnification (one dot is displayed as one dot), and two-times magnification (one dot is displayed as four dots).

The 8x8 dot-pattern is displayed as 8 dots x 8 dots on a one-time magnification. If the pattern is magnified by two times, it is displayed as 16 dots x 16 dots, but the resolution is not improved. Changing the pattern size does not change the resolution.

The sprite resolution and magnification ratio must be applied to all sprites and, therefore, cannot be specified for single sprites.

Subroutine calling sequence

Subroutine name: MAGFY

Address: 045CH

Function:

Magnifies the sprite.

**Contents**

Input

Register

Acc

Kinds of magnification

|  |  |  |
| --- | --- | --- |
|  | Resolution | Mag. ratio |
| 0 | 8x8 | **X 1** |
| 1 | 8x8 | **X 1** |
| 2 | **16 x 16** | x 2 |
| 3 | **16 x 16** | x 2 |

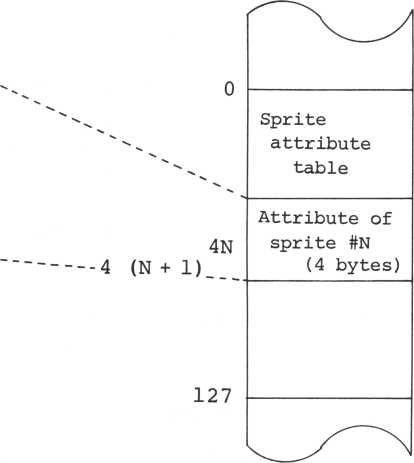
This subroutine determines the resolution and size of the sprite.

1. 3 Contents of the sprite attribute table

To display the sprite on the screen, the sprite position, color, code, etc. must be set. They are stared in the sprite attribute table in VRAM.

|  |  |  |
| --- | --- | --- |
| **Vertical**  **position** | | **(Y)** |
| **Horizontal position (X)** | | |
| **Code no.** | | |
| **EC** | **0 0 0** | **Color** |

The sprite attribute table consists of four bytes per sprite and is set in the following order: vertical position, horizontal position, code number, and color.



EC is the "early clock" bit and is a flag for shifting a sprite to the left by 32 dots.

The sprite attribute table consists of four bytes per sprite, (128 bytes for the maximum 32 sprites). Positions of the sprite attribute table correspond to the sprite numbers. The sprite numbers coincide with the sprite display priority. 0 is the highest priority.

Subroutine calling sequence

Subroutine name 2 STSCOD

|  |  |
| --- | --- |
| Address: | 0454H |
| Function 2 | Sets the sprite code. |
| Input  Registers | Contents |
| Acc | Sprite no. |
| C | Graphics code no. |

Resister save: B,C,D,E,X,Y

This subroutine is the graphics code handling routine for sprites. When the sprite magnification is two,by setting the graphics code for the upper left corner in register C, the remaining three patterns are set in the next three locations.

Subroutine name: STSCOL

|  |  |
| --- | --- |
| Address: | 0445H |
| Function: | Sets the sprite color. |
| Input  Registers | Contents |
| Acc | Sprite no. |
| B | Color code |

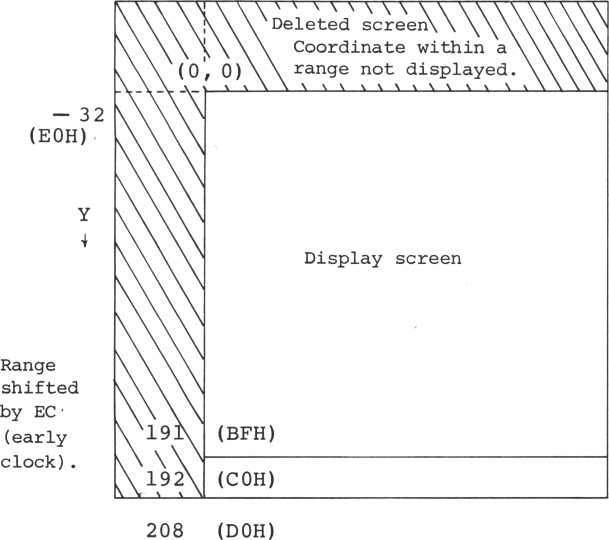
Register save: B,C,D,E,X,Y

This subroutine sets the color of a sprite. Only the least-significant four bits are effective for the color code.

1. 4 Sprite position (physical position and logical position)

The sprite position is indicated by a dot position on the screen with the upper left corner of the sprite taken as the origin.

— 48 (DOH) x -> 255



Dots on the display screen are arranged as 192 vertical (Y) dots and 256 horizontal (X) dots. The screen between Y=192 (COH) and Y=255 (FFH) cannot be seen. Exceeding 255 causes a sprite deleted downwards to appear.

Setting EC (early clock) at position X shifts the position to the left by 32 dots; the position can be hidden from the screen.

Physical position and logical position

This computer handles sprites with a logical binary position (with a two-byte code for software manipulation), computes the physical position (dot position on the screen), and outputs the results to the sprite attribute table.

Users can move the sprite without considering the early clock or position overflow.

Subroutine calling sequence

Subroutine name: MVSPA

|  |  |
| --- | --- |
| Address: | 03CEH |
| Function: | Changes the sprite position. |
| Input  Registers | Contents |
| Acc | Sprite no. |
| D,E | Position (logical) Y |
| H ,L | " X |
| Output  Registers | Contents |
| E | Position (physical) Y |
| D | " X |
| B | Early clock bit (MSB) |

Register save: H,L,X,Y

Other conditions: Cy=l The sprite disappeared

from the screen. The sprite position exceeded the support range.

Subroutine name: GTSPOS

|  |  |
| --- | --- |
| Address: | 042BH |
| Function: | Gets the current sprite |
|  | position. |
| Input  Register | Contents |
| Acc | Sprite no. |
| Output  Registers | Contents |
| B | Sprite color code |
| C | " graphics code |
| D,E | Current position (logical) Y |
| H,L | " X |

Register save: X,Y

Other conditions: Cy=l The sprite is out of

|  |  |
| --- | --- |
|  | the support range. |
| Remarks: | Even if the sprite position |

is out of the support range,

|  |  |
| --- | --- |
|  | the color code and graphic |
|  | code are given. |

This subroutine gets all sprite attributes.

Subroutine name: GTSPLC

|  |  |
| --- | --- |
| Address: | 03EEH |
| Function: | Gets the logical sprite position. |
| Input  Registers | Contents |
| B | Early clock bit (MSB) |
| C | Physical position X |
| E | " Y |
| Output  Registers | Contents |
| D**,** E | Logical position Y |
| H ,L | " X |

Register save: B,C,X,Y

Other condition: Cy=l The sprite is out of

the support range.

This subroutine provides the physical sprite position and computes the logical position.

Subroutine name: GTSPPC

|  |  |
| --- | --- |
| Address: | 0377H |
| Function: | Gets the physical sprite |
|  | position. |
| Input  Registers | Contents |
| D, E | Logical position Y |
| H ,L | " X |
| Output  Registers | Contents |
| D | Physical position X |
| E | " . Y |
| B | Early clock bit (MSB) |

Register save: X,Y

Other conditions: Cy=l The sprite is out of

|  |  |
| --- | --- |
|  | the support range.  The sprite disappeared from the screen. |

This subroutine gives the logical sprite position and computes the physical position.

The sprite is moved by rewriting the sprite position.

Subroutine calling sequence

Subroutine name: MVSPR

|  |  |
| --- | --- |
| Address: | 040BH |
| Function: | Relative move of the sprite |
| Input  Registers | Contents |
| Acc | Sprite no. |
| B | Moving vector (X direction) |
| C | " (Y direction) |
| Output  Registers | Contents |
| D ,E | Position after the move Y |
| H,L | " X |

Register save: X,Y

Other conditions: Cy=l The sprite disappeared

from the screen.

This subroutine gives the moving vector and moves the sprite. The moving vector is indicated in dot increment. A binary number with a one-byte code is used for the vector.

Subroutine name: ADDVCT

|  |  |
| --- | --- |
| Address: | 041BH |
| Function: | Moving vector addition |
| Input  Registers | Contents |
| H,L | Vector to be added X |
| D,E | " Y |
| B | Vector to add X |
| C | " Y |
| Output  Registers | Contents |
| H,L | Added result X |
| D ,E | " Y |

Register save: Acc,X,Y

This subroutine adds the moving vector of the sprite The "vector to be added" of DE and HL indicates the sprite move volume in dot increments.

Subroutine name: GTSTEP

|  |  |
| --- | --- |
| Address: | 049FH |
| Function: | Finds a moving vector |
|  | near the target. |
| Input  Registers | Contents |
| B | Target position X |
| C | " Y |
| H,L | Current sprite position X |
| D,E | " Y |
| Output  Registers | Contents |
| B | Moving vector X |
| C | " Y |

Register save: Acc,X,Y

Other conditions: Cy=l The target is too far

away (the distance to the target should be

within 255 dots).

The moving vector

reached the target.

This subroutine finds a moving vector for the sprite near the target. The size of the main direction (whichever is longer - the X distance or the Y distance) of the moving vector is determined by the number of sprite move steps in the event information table.

It easy to advance the sprite to a specified position by using GTSTEP and MVSPR.

To delete the sprite, move the sprite to a position which is off the screen.

Subroutine calling sequence

Subroutine name: ERSSPR

|  |  |
| --- | --- |
| Address: | 03CCH |
| Function: | Deletes the sprite. |
| Input  Register | Contents |
| Acc | Sprite no. |

Register save: X,Y

This subroutine moves the sprite to a fixed deletion position. The sprite position runs out of the system support range.

Subroutine calling sequence

Subroutine name: DELSPR

|  |  |
| --- | --- |
| Address: | 03C5H |
| Function: | Deletes the sprite. |
| Input  Register | Contents |
| Acc | Sprite no. |

Register save: B,C,D,E,X,Y

This subroutine gives DOH to the sprite position Y.

As a result, positions with a lower priority than the specified sprite cannot be displayed. DOH is a hardware delete code.

|  |  |
| --- | --- |
| Subroutine name 2 | ERSPRA |
| Address: | 1387H |
| Function: | Deletes all sprites. |
| Register save: | X,Y |

This subroutine deletes all the sprites. Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine name: | CLRSS |
| Address: | 13 7EH |
| Function: | Clears the screen and |
|  | deletes the sprite. |
| Register save: | X,Y |

This subroutine clears the screen and deletes the sprite.

The VDP provides the following sprite information to the CPU (the sprite status (SPSTUS) within the event control table):

LSB

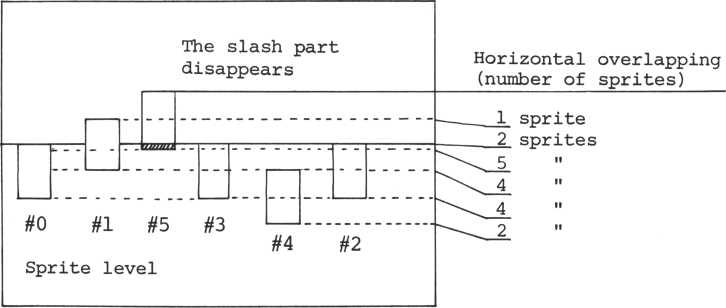
**76543210**

Fifth sprite no.

Sprite coincidence flag

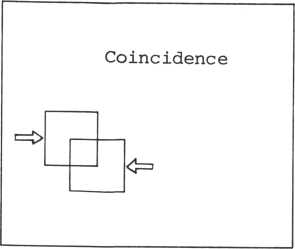
Fifth sprite flag

When more than four sprites are arranged on the same line due to a hardware restriction, VDP TSM9918A delets overlapping part of the fifth sprite. As illustrated below, the overlapping part beyond the fifth sprite on the lowest level disappears. The status is given by bits 6 and 0 ~ 4 of STATUS.



Bit 5 is the flag that shows sprite coincidence when set

to 1. Since the contending sprite numbers are not given, the user must determine them himself.



Chapter 5 Event Handler

CONTENTS

1. [What is an event? 5-1](#bookmark60)
2. [Event control flag 5-5](#bookmark62)
3. [System clock 5-6](#bookmark63)
4. Alarm processing 5-8
5. Time processing routine 5-9
6. Up counter 5-10
7. Down counter 5-12
8. [User events 5-14](#bookmark64)
9. 1 What is an event?

5-1-1 Definition and types of events

An "event" is the processing for an interrupt after a fixed interval in response to an interrupt of the CTC.

There are two types of events: a user event is initialized when the user program is run, and a system event is supported by the system from startup and is always running.

5-1-2 System events and user events

A user event can be determined by the user as required, but care should be exercised during execution of the user event. A user can define up to 40 user events.

A system event is supported by the system and can be utilized by the user according to the system procedure.

5-1-3 Types of system events

A system event can control the following functions:

1. System clock
2. Up counter and down counter
3. Interrupt of keyboard and joypad
4. Reset and halt key support

**Subroutine name**

**CTCINT**

Address

**01C2H**

Initializes CTC

Function:

Register save

**X,Y**

This sub-routine issues a control instruction to each CTC channel and initializes the operation. Initialization data for CTC channels #0 - 3 is as follows:

CTC initialization data

CTC initialization data

Channel **#0**

D7 ... INTEN (interrupt enable)

D6 ... Counter mode D5 ... X

D4 ... Rising trigger D3 ... X

D2 ... Presence of constant setting D1 ... Reset DO ... 1

Channel #1 D7 ... INTEN D6 ... Timer mode

D5 ... Prescaler **X** 256 D4 ... Rising trigger D3 ... Automatic initialization D2 ... Presence of constant setting

Dl ... Reset DO ... 1

Channel **#2** D7 ... INTDS D6 ... Counter mode D5 ... X

D4 ... Rising trigger D3 ... X

D2 ... Presence of constant setting Dl ... Reset DO ... 1

Channel **#2 Dl** ... INTEN D6 ... Counter mode D5 ... X

D4 ... Rising trigger D3 ... X

D2 ... Presence of constant setting Dl ... Reset DO ... 1

|  |  |
| --- | --- |
| **Channel No.** | **Data (hexadecimal)** |
| **#0** | **C701** |
| **#1** | **A70E** |
| **#2** | **5717** |
| **#3** | **C701** |

For the details of set values, refer to the Z80 CTC Technical Manual.

CTC3SP

01DFH

Subroutine name: Address:

Function:

Register save:

Supports an interrupt of CTC channel 3.

•p

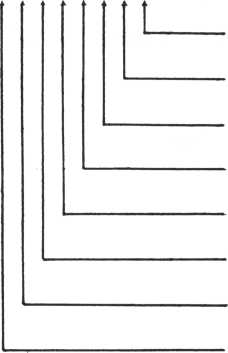
Support is provided by the system event support processing and event flag status, functions of which are listed below.

1. Sprite interrupt processing
2. System clock
3. Up counter
4. Down counter
5. Reset and halt key
6. Joypad direction switch
7. Stop of click and bell ringing
8. User event support
9. 2 Event control flag

The event control flag is used to set the control status of system events and user events.

76543210

Synchronous sprite display System clock switch Up counter switch Down counter switch Joystick switch Keyboard switch User event switch Down counter end flag



**(1**

**(1**

(1

**(1**

**(1**

**Cl**

**(1**

**(1**

**ON)**

**ON)**

**ON)**

**ON)**

**ON)**

**ON)**

**ON)**

**ON)**

1. 3 System Clock

This computer has its own "clock". The system clock is set to 00:00:00 when power is turned on. The system clock is accurate to about 0.1%.

This clock performs two functions: timed interrupt every hour and alarm interrupt.

Subroutine calling sequence

Subroutine name: STSCLK

|  |  |
| --- | --- |
| Address: | 02CFH |
| Function: | Sets time of the system clock |
| Input  Registers | Contents |
| Acc | Hour data |
| H | Minute data |
| L | Second data |

Register save: Acc,B,C,D,E,H,L,X,Y

Subroutine calling sequence

Subroutine name: GTSCLK

|  |  |
| --- | --- |
| Address: | 02DBH |
| Function: | Reads time from the system |
|  | clock |
| Output  Registers | Contents |
| Acc | Hour data |
| H | Minute data |
| L | Second data |

Register save: B,C,D,E,X,Y

the system clock time. The alarm is set directly as shown below:

. ALMTMM ... Alarm time (minute)

} One byte each

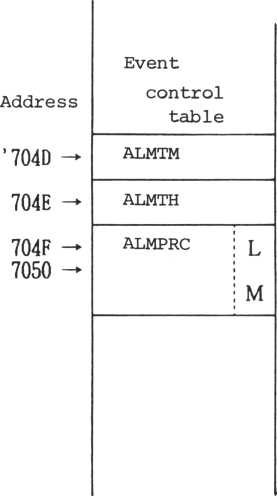
. ALMTMM ... " (hour)

Note that no interrupt is allowed while these two bytes are set.

Alarm processing is used to initialize an interrupt at a preset time.

To set the alarm, store the alarm time at ALMTM and ALMTH in the event control table and store the head address of the alarm routine at ALMPRC.

Alarm time (minute)



Alarm time (hour)

Head address of alarm routine

The time processing routine performs initialization at 00:00 each hour. To execute time processing, store the head address of the routine each hour at EVHPRC in the event control table.

The up counter is a two-byte counter with a minimum count cycle of 1/60 of a second. The count cycle is determined by UPCTBI (standard up counter time) in the event control table.

Count cycle = (standard up counter time) x 1/60 sec.

The default value of the count cycle is one second, and the maximum counter value is 65,535.

Subroutine calling sequence

Subroutine name: STRTUC

Address: 02FBH

Function: Initializes the up counter.

**Input**

**Registers Contents**

Acc Standard up counter time

Register save: Acc,B,C,X,Y

Other conditions: Cy=l The up counter is

operating.

This subroutine starts the up counter from 0. If the up counter has already started, an error occurs, but this does not affect counters already operating.

The count cycle depends on the standard time set in (Acc).

|  |  |
| --- | --- |
| Subroutine name: | STOPUC |
| Address: | 0313H |
| Function: | Stops the up counter |
| Register save: | Acc,B,C,D,E,X,Y |

This subroutine stops the up counter. If the counter has already been stopped, nothing happens. When the counter stops, a reset does not occur but the counter value and standard time are stopped at their current values.

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine name: | RSTRUC |
| Address: | 030DH |
| Function: | Reinitializes the up counter |

Register save: Acc,B,C,D,E,X,Y

This subroutine restarts a stopped up counter from the current initial value and standard time, without change.

The down counter is a two-byte counter with a minimum count cycle of 1/60 second.

When the down counter value is 0, the down counter end flag (bit 7) (one of the event control flags) is turned ON (1), and the count is stopped. The count cycle is determined by the standard down counter time like that for the up counter:

Count cycle = (standard down counter time) x 1/60 sec The count cycle default value is one second.

This down counter is used as a keyboard timer to count the keyboard time-out. Be careful of this correspondence with the key input instruction when operating this counter.

Subroutine calling sequence

Subroutine name: STRTDC

0319H

**Address:**

**Function:**

**Input**

**Registers**

**Acc**

**H, L**

**Register save**

Initializes the down counter **Contents**

Standard down counter time Initial value : B,C,D,E,X,Y

Cy=l The down counter is  
operating.

**Other conditions:**

registers (H) and (L) to 0 and sets the (Acc) value to the standard interval.

If the down counter is already operating, an error occurs, but other operating counters are not affected.

When the down counter starts, the down counter switch is ON (1) and the down counter end flag is set to 0.

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine name: | STOPDC |
| Address: | 0340H |
| Function: | Stops the down counter |
| Register save: | Acc,B,C,D,E,H,L,X,Y |

This subroutine stops the down counter. The counter value and standard time are stopped at their current values.

Subroutine calling sequence

|  |  |
| --- | --- |
| Subroutine name: | RSTRDC |
| Address: | 0336H |
| Function: | Restarts the down counter |
| Register save: | Acc,B,C,D,E,X,Y |
| Other conditions: | Cy=l The down counter has |
|  | stopped |

This subroutine restarts a stopped down counter. If the down counter end flag has already been set to 0, an error occurs. Otherwise the count restarts with the previous down counter values.

5-8 User events

A user event processes user-defined interrupts.

Control of a user event (start, stop, etc.) is managed by M5XEV.SR (the external monitor handler), but it is possible to use a user event without these routines.

1. User event save

To save a user event:

1. Set the maximum number of saved events in the event control table and the head address of the event information table, and turn on the user event switch of the event control flag.
2. Write the required six bytes of information for each event into a user-defined event information table.
3. User event start and stop

System support is available by setting the event

wait flag of the event information table to 0.

Setting it to 1 causes the event wait status, and

the event stops.

|  |  |
| --- | --- |
|  |  |
| **Number of saved events** | |
| (N) |  |
| **Event** | L |
| **information table** |
| head  address | H |

ui

<D

-P

>1

XI

**Event information table**

**Support**

**delay**

F

**time**

**Support**

**interval**

**User event processing routine**

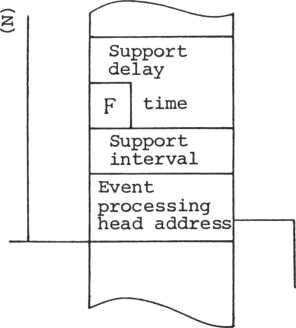
**Event**

**processing head address**

User event processing

V0

**X**



routine

F: Event wait flag (N): 40 or less



The support delay time is determined by a 15-bits counter.

|  |  |  |
| --- | --- | --- |
| s  F | **upport delay time** | |
| **Support**  **interval** | | i |
| **Support**  **interval** | | w |
| **Event**  **processing head address** | | L  H |

It is the interval from the event save to the first support time.

The range of the support delay time is:

0 ^ T ^ 546 seconds

L Event wait flag

**T = (support delay time) x 1/60 sec**

1. = Support request
2. = Support wait

The support interval is measured in units of the CTC interrupt interval (1/60 second each).

The support interval (T) expression is:

T = (Support interval) x 1/60 sec

Event processing head address

When the user event processing time exceeds the interrupt interval, a run error may occur.

ETREIT

0356H

**Subroutine name: Address:**

**Function:**

Saves the event information table

**Contents**

**Input**

**Registers**

**Acc**

**H,L**

Number of saved events

Event information table head address

Register save: Acc,B,C,D,E,X,Y

Other conditions: Cy=l The number of saved

events is inadequate.

This subroutine sets the maximum number of saved events and the event information table head address and turns on the user event switch of the event control flag. If the number of saved events exceeds 40, an error occurs.

Chapter 6 Saving to and Loading from Cassette Storage

CONTENTS

1. Output format 6-1
2. File format 6-2
3. [Baud rate 6-5](#bookmark72)
4. Read/write of file ID block 6-6
5. Write/read of block data 6-7
6. [Combined read/write 6-14](#bookmark79)
7. [Motc>r ON/OFF 6-17](#bookmark80)

o Recording system: Pulse-width modulation,

FSK method (0 + f, 1 -\* 2f)

o Transfer rate: 1600 - 3200 (bps) optional

o Waveform

0



|  |  |
| --- | --- |
|  |  |
| 2T | T |

o 1600 bps

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0: | f = 1200 | Hz | 2T = 833 | ysec |
| 1: | 2f = 2400 | Hz | T = 417 | ysec |

**1. ID format**

0

1

2

3

4

5

6

7

8

9

1. 11

12

13

14

15

16

17

18

19

1. 21 22

23

24

25

26

27

28

29

30

|  |  |
| --- | --- |
| **File attribute** | |
| **File name** |  |
| **(9 characters)** | |
| **Loading head** | L |
| **address** | H |
| **Data (program)** | L |
| **Size**  **(in bytes)** | H |
| **Program start** | L |
| **address** | H |
| **Expansion**  **attribute** | |
| **Blank area** |  |
| **(14 bytes)** |  |

0

1

2

3

4

5

6

7

0

**Data**

**Loading only**

**Consecutive**

**data**

**CPU memory**

1

**Execution format (machine language)**

**Auto-start**

**Data file**

**VDP memory**

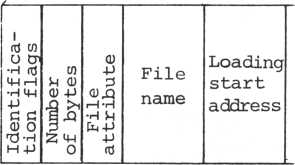
**Without pvr^n- With expansion sion attribute attribute**

**Language**

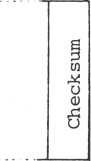
**classification (2) (2) (2)**

|  |  |
| --- | --- |
| Language | classification |
| = 0 |  |
| 1 | Basic-I source |
| 2 | Falc |
| 3 | Basic-G |
| 4 | Basic-F |
| 5 | reserved |
| 6 | reserved |
| 7 | reserved |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1**  **u** | **Tape** | **File** | **I** | **Data** |  | **I** | **Data** | **I** | **Data** | **I**  **B**  **G** | **EOF** |
| **lan**  **run** | **mark** | **ID** | **B**  **G** | **block** |  | **B**  **G** | **Block** | **B**  **G** | **block** | **block** |
| **r** |  |  |  |  |  |  |  |  |  |  |  |



**Blank run:**



|  |  |  |  |
| --- | --- | --- | --- |
| i  rtf to O | to |  | g |
| \*H rtf | **U** 0 |  | to |
| m r—1 | o -p | **Data** |  |
| •h m | **A** >i |  | U |
| -P | sn |  | 0 |
| Iden  tion | 0 |  | **6** |

Non-recording period (about four seconds, due to a tape recorder with a head-out feature)

8,000 marks

**Tape marks: File ID:**

**IBG:**

Refer to preceding page

Non-consecutive data:

3,600 marks

Consecutive data: 512 marks

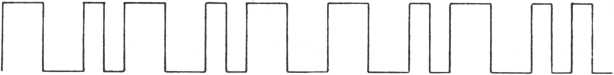
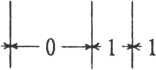
|  |  |  |
| --- | --- | --- |
| Identification flags: | 'H' | (48H) = File ID |
|  | 'D' | (44H) = Data block |
|  | 'E ' | (45H) = EOF block |
|  | i p i | (46H) = File block |

The EOF block is added only when non-consecutive data is output.

When one byte of data is sent, the data is sent as 10 bits by adding a start bit and a stop bit.

Start bit = 0 Stop bit = 1

Example A5H



|  |  |
| --- | --- |
|  | , J |
| **Start**  **bit** | **LSB** |

MSB

Stop bit

1. 3 Baud rate

The baud rate is given by the following expression (at the save time).

Buad rate = {(d^ + n) 1 + 2 (dQ + 2n) 1} 7 3

|  |  |  |
| --- | --- | --- |
| Where, | dl | = 35.36 x 10 6 |
|  |  | -6 |
|  | do | = 49.71 x 10 |
|  | -6 |
|  | n | = N x 8.929 x 10 D |
|  | N | = loop counter (STDLY value, in system table) |

Initial value = 33

At the load time the baud rate is determined by the reader section (a section of more than 256 continuous bits).

Hence, the user need not specify the baud rate, but there is a limitation - with a lower limit of 1170 bps and an upper limit of about 4000 bps. The general tape recorder test value indicates a lower limit of 1170 bps and upper, limit of 2830 bps.

Subroutine calling sequence

Subroutine name: RDFID

|  |  |
| --- | --- |
| Address: | 1598H |
| Function: | Reads the file ID block |
| Input  Registers | Contents |
| H,L | Head address of file ID input buffer |

Register save: X,Y

Other conditions: Cy=l The break key was pressed

Checksum error

Subroutine calling sequence

Subroutine name: WTFID

|  |  |
| --- | --- |
| Address: | 15C3H |
| Function: | Writes out the file ID block |
| Input  Registers | Contents |
| H,L | Head address of file ID table |

Register save: X,Y

Other conditions: Cy=l The break key was pressed

Subroutine calling sequence

Subroutine name: WATBL

|  |  |
| --- | --- |
| Address: | 15EBH |
| Function: | Outputs block data (data block) |
| Input  Registers | Contents |
| B | Number of output bytes |
| C | IBG length |
| H ,L | Output buffer head address |
| Output  Registers | Contents |
| H, L | Data address last output + 1 |

Register save: X,Y

Other conditions: The break key was pressed.

This subroutine outputs IBG and output buffer data equivalent to the specified number of bytes.

When bit 0 of the system table and ACMT table is 0 , the main memory is used as the output buffer, when 1, the VDP memory is used.

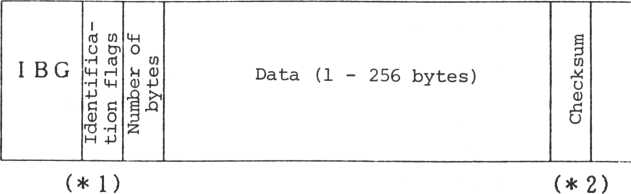
Subroutine name: WATBLF

|  |  |
| --- | --- |
| Address: | 15E8H |
| Function: | Outputs a block of data |
|  | (file block) |
| Input  Registers | Contents |
| B | Number of bytes output |
| C | IBG length |
| H, L | Output buffer head address |
| Output  Registers | Contents |
| H ,L | Data address last output + 1 |

Register save: X,Y

Other conditions: Cy=l The break key was pressed

This subroutine basically performs the same processing as subroutine WATBL except that data is output in a block.



\*1 Outputs the IBG length (C register value) x 256 bytes' wave 1.

Ex. IBG length (number of pulses) = 8, at 1600 bps (1 = 2400 Hz)

IBG = 8 \*256 = 2048 pulses — \_2048 \_ seconds

2400

\*2 One-byte unsigned data without regard to carry (1 - 256 bytes).

Subroutine name: RATBL

**Address:**

**Function:**

**Input**

**Registers**

**H,L**

**Output**

**Registers**

**1650H**

**Reads the data block**

**Contents**

**Input buffer head address**

**Contents**

Data address last input + 1

**H,L**

Register save: X, Y

Other conditions: Cy=l The break key was pressed

Checksum error

An illegal block was read The buffer was filled.

The EOF block was read.

This subroutine provides data blocks and outputs data to the input buffer (access is required during the IBG); it also selects the input buffer according to the system table and buffer select switch of the ACMT system flag.

Input buffer head address

Input data (1 - 256 bytes)

(H) and (L) value at normal end time (address)

Subroutine name: RATBLF

|  |  |
| --- | --- |
| Address: | 164DH |
| Function: | Reads the file block |
| Input  Registers | Contents |
| H,L | Input buffer head address |
| Outpur  Registers | Contents |
| H ,L | Data address last input + 1 |

Register save: X,Y

Other conditions: 0^=1 T^e kreak key was Pressed

|  |  |
| --- | --- |
|  | Checksum error |
|  | An illegal block was read. |
|  | The buffer was filled. |
|  | The EOF block was read. |

This subroutine outputs block data to the input buffer and various data to the file block.

**Subroutine name: CPFNM**

**Address:**

**Function:**

**Input**

**Registers**

**D, E**

**H,L**

**1765H**

**Compares data Contents**

**Data head address to be compared Data head address to be compared**

**Register save: Other conditions: Remarks:**

**B,C,X,Y**

**Cy=l No match**

**This processing is used by the next CPFNM.**

**This subroutine compares data at the address in the DE register to data at the address in the HL register.**

**If the two pieces of data do not match, it sets a carrier. There is no limitation on the lengths of data that are compared. When a terminator (null character) is encountered, the data is assumed to be complete.**

**When a ? (3FH) is encountered, the comparison stops.**

(D) (E)

DATA 1

♦ (H) (L)

DATA 2

NULL

NULL

Subroutine name: RFIDC

|  |  |
| --- | --- |
| Address: | 1587H |
| Function: | Reads the file ID and compares |
|  | the file names. |
| Input  Registers | Contents |
| H,L | Head address of file ID input buffer |
| D,E | Head address of file names to be |
|  | searched. |

Register save: X,Y

Other conditions: Cy=l The break key was pressed

Checksum error.

The file names do not match.

Subroutine calling sequence

**6-6 Combined read/write**

Subroutine name: BSAVE

|  |  |
| --- | --- |
| Address: | 1529H |
| Functions: | Saves continuous data. |
| Input  Registers | Contents |
| H,L | Head address of file ID table |

Register save: X,Y

Other conditions: Cy=l The break key was pressed

|  |  |
| --- | --- |
| Remarks: | This subroutine is used by |
|  | each processing described so |
|  | far; it saves data according |
|  | to information specified within the file. |

Subroutine calling sequence

Subroutine name: BSAVD

|  |  |
| --- | --- |
| Address: | 1563H |
| Function: | Divides data into blocks and |
|  | saves it. |
| Input  Registers | Contents |
| H, L | Output data head address |
| B ,C | Number of bytes output |

Resister save: X,Y

Other conditions: Cy=l The break key was pressed

|  |  |
| --- | --- |
| Remarks: | Data is divided into 256-byte |
|  | blocks and output. |

Subroutine name: BLODD

|  |  |
| --- | --- |
| Address: | 1579H |
| Function: | Loads one block of data |
| Input  Registers | Contents |
| H,L | Load buffer head address |
| B,C | Number of bytes input |

Register save: X,Y

Other conditions: Cy=l The break key was pressed

|  |  |
| --- | --- |
|  | Checksum error |
|  | The input buffer was |
|  | filled. |
|  | An illegal block was read |
|  | The EOF block was read. |
| Remarks: | Data is divided into 256-byte |
|  | blocks and input. |

|  |  |
| --- | --- |
| Subroutine | name: BLOAD |
| Address: | 153BH |
| Function: | Reads continuous data. |
| Input  Registers | Contents |
| Acc | Loading mode: |
|  | 0 Loading |
|  | 1 Verify |
| H,L | Head address of file ID table |

Register save: • X,Y

Other conditions: Cy=l The break key was pressed.

|  |  |
| --- | --- |
|  | Checksum error |
|  | The file name is different |
|  | An illegal block was read. |
| Remarks: | Data is loaded or verified |
|  | according to information speci |
|  | fied by the file ID. |

**6-7 Motor ON/OFF**

|  |  |
| --- | --- |
| Subroutine name: | MTRON |
| Address: | 1776H |
| Function: | Turns on the ACMT remote switch. |
| Register save: | All registers |
| This subroutine turns | on the remote switch of the ACMT |
| deck. |  |

**Subroutine calling sequence**

|  |  |
| --- | --- |
| Subroutine name: | MTROF |
| Address: | 177EH |
| Function: | Turns off the ACMT remote switch. |
| Register save: | All registers |

**This subroutine turns off the remote switch of the ACMT deck.**

Chapter 7 Sound Generator Handler

CONTENTS

1. [The sound generator 7\_1](#bookmark82)
2. [SML interpreter 7-7](#bookmark85)
3. Interrupt processing by SML\* 7-18
4. [How to use the noise generator 7-19](#bookmark93)

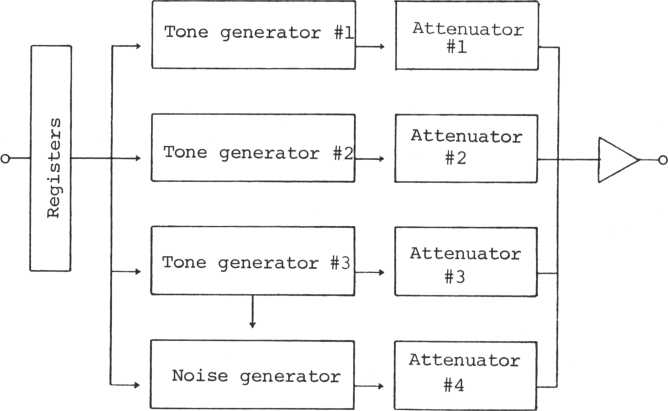
\* SML Sord Music Language

1. 1 The sound generator

The sound generator (SN76489AN) consists of three tone generators and one noise generator. The block diagram is given below.

Input

Output



7-1-1 Frequency set format (two-byte output)

1st byte

**MSB**

|  |  |  |
| --- | --- | --- |
| 7 6 5 4 3 2 1 0 | | |
|  | \2 Ri Ro | 3 2 1 0 |
| 1 | Reg.Adr. | Da ta |

**(Lower 4 bits)**

2**nd byte**

|  |  |  |
| --- | --- | --- |
| MSB |  | LSB |
| 7 | 6 | 5 4 3 2 1 0 |
|  | 9 8 7 6 5 4 | |
| 0 | X | Da ta |

**(Upper 6 bits)**

9 8 7 6 5 4

3 2 10

Reg.Adr.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| R 2 | R i | Ro | **Register no.** | |
| 0 | 0 | 0 | **Tone generator** | 1 |
| 0 | 1 | 0 | ii | 2 |
| 1 | 0 | 0 | ii | 3 |
| Frequency | | | calculation |  |

**Data n**

**(10-bit frequency divisor)**

f[Hz] = N/32n f: Output frequency

N: Clock input frequency (3.579545 MHz)

n: 10-bit frequency division ratio

|  |  |  |
| --- | --- | --- |
| 7 6 5 4 3 2 1 0 | | |
|  |  |  |
| 1 | R2 R i Ro  j | Delta j |

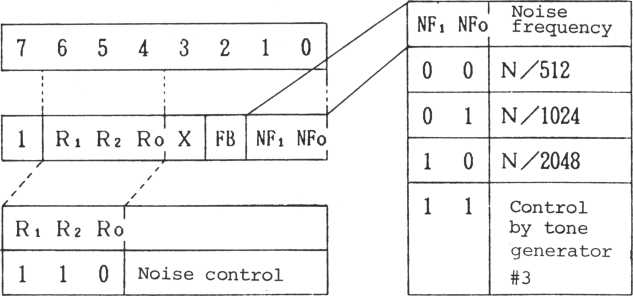
**/ / / /**

|  |  |  |  |
| --- | --- | --- | --- |
| **L**  Rz | R i | Ro | f 1 |
| 0 | 0 | 1 | Tone attenuation 1 |
| 0 | 1 | 1 | " 2 |
| 1 | 0 | 1 | " 3 |
| 1 | l | 1 | Noise attenuation |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A3 | Az | At | —  Ao | **1 Attenuation . Volume** |
| 0 | 0 | 0 | 0 | 2 |
| 0 | 0 | 1 | 0 | 4 |
| 0 | 1 | 0 | 0 | 7 |
| 1 | 0 | 0 | 0 | 12.5 |
| 1 | 1 | 1 | 1 | OFF |

**Note:**

**Changing the attenuation at every fixed time enables an envelope to be generated.**



|  |  |
| --- | --- |
| FB | **Noise mode** |
| 0 | **Synchronous**  **noise** |
| 1 | **White noise** |

7-1-4 Precaution for noise generator operation

The output of the noise generator consists of white noise and synchronous noise. The noise frequency is controlled by two methods: fixed control (three kinds N/512, N/1024, and N/2048) and control under tone generator #3.

Items that can be controlled by tone generator #3

1. The kind of noises.
2. The noise volumes.
3. The noise frequency (2 bytes).

Only the above information is required (OUT instruction) .

Changing the noise volume and frequency can affect the output sound remarkably.

Ex: Gunshot

Change the frequency register set value of the tone f3 to 0 - 15 and the noise attenuator to 0 - 15.

|  |  |
| --- | --- |
| **Sound scale** | **Data** |
| **Octave 3 do** | **855** |
| „ , **# " do** | **807** |
| **" re** | **762** |
| **„ # " re** | **719** |
| **" mi** | **679** |
| **" fa** | **641** |
| II JT #  **" fa** | **605** |
| **" sol** | **571** |
| II t #  **" sol** | **539** |
| **" la** | **508** |
| **la#** | **480** |
| **" si** | **453** |

**\* Any tone in octave 4 can be the corresponding tone data**

**found multiplying for octave 3 by 1/2.**

7-2 SML interpreter

The SML interpreter interpretes eight commands to express the sound scale, duration, tempo, etc. It controls the SGC and timer.

This interpreter is executed by interrupt processing, with the interrupt cycle set by the tempo command.

. Actual case of interrupt



T^ = 1 msec

T = 1

2 Tempo counter

6 0

Tempo counter = — **tz x 1000**

^ Tempo x 16

= Duration of a 1/64 note.

The SML handles following eight commands. Number of

**Type bytes Function**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1-1 | 1 | Specifies the interval and rest within the same octave. | | |
| -2 | 2 | Specifies duration. | the | above + sound |
| II-l | 2 | Specifies number. | the | interval by a |
| -2 | 3 | Specifies duration. | the | above + sound |
| III | 1 | Specifies the transposition, | | octave or |
| IV | 1 | Specifies | the | sound volume. |
| V | 1 | Specifies | the | envelope form. |
| VI | 2 | Specifies | the | hold time no. |
| VII | 2 | Specifies | the | sound duration |
| VIII | 2 | Specifies | the | temp. |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | 0 | 0 |  | **.interval and rest'** | **When** | **bit 7** | **= 1** |
| **n** | **2nd** | **= sound** | **duration** |
| **7** | **6** | **5** | **4** | **3** | **2 1 0** |  | **data** |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Directly specifies the interval by a number** | | | | | |
| **0** | **0** | **0** |  | | **When 2nd, bit 7=1 2nd: Interval data** |
| **7 6 5 4 3 2 1 0 Specifies the octave or tran** | | | | | **3rd: Sound duration data**  **sposition**  **When bit 7=1 n = transposition**  **i** |
| **0 | 0** | | **1** | **0** | **n (octave)** |
| **7 6 5 4 3 2 1 0 Specifies the sound volume** | | | | |
| 0 | **0** | **1** | **1** | **n (sound volume)** |  |
| **7 6 5 4 3 2 1 i Specifies the envelope form** | | | | | **0**  **(no.)** |
| **0** | **1** | **0** | **0** | 0 **J n (envelope no.)** | |
| **7 6 5 4 3 2 1 0**  **Specifies the hold time no.** | | | | | |
| 0 | **1** | 0 | **1 J n (hold time no.)** | | |
| **7 6 5 4 3 2 1**  **Specifies the sound duration** | | | | | 0 |
| 0 | **1** | **1** | 0 |  | **2nd: Sound duration**  **data** |
| **7 6 5 4 3 2 1 0**  **Specifies the tempo** | | | | | |
| **my**  **i**0**11** | | **1** | **l** |  | **2nd: Tempo data** |

**II.**

**III.**

**IV.**

**V.**

**VI.**

**VII.**

**VIII.**

7 6 5 4 3 2 1 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **7** | **6** | **5** | **4** | **3 2 10** |
| **1st.** | **0** | **0** | **0** | **0** | . **interval.^ n (rest** > |

• 0Sn£12

|  |  |  |  |
| --- | --- | --- | --- |
| **n = 0** | **R** | **Rest** |  |
| **1** | **C** | **do** |  |
| **2** | **C+, D-** | **do#,** | **b**  **re** |
| **3** | **D** | **re** |  |
| **4** | **D+, E-** |  | **.b**  **mi** |
| **5** | **E** | **mi** |  |
| **6** | **F** | **fa** |  |
| **7** | **F +, G-** | **fa#,** | **solb** |
| **8** | **G** | **sol** |  |
| **9** | **G +, A —** | **sol#,** | **lab** |
| **10** | **A** | **la** |  |
| **11** | **A + B-** | **la#** | **si^** |
| **12** | **B** | **si** |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **7** | **6** | **5** | **4** | **3 2 10** |
| **l** | **0** | **0** | **0** | **.interval,. n (rest )** |
| **7** | **6** | **5** | **4** | **n = same as I**  **3 2 10** |
| **0** | **m** | **(sound** | | **duration)** |

• 1 Sm^64

|  |  |
| --- | --- |
| **64** | **\*** 1 **'** |
| **56** | **’ 2.,** |
| **48** | ’ 2. |
| 32 | **’ 2 1** |
| 24 | **\*** 4. |
| 16 | 1 4 **'** |
| 4 | **’** 16’ |
| 2 | ’32’ |
| 1 | 1 64 **\*** |
| 0- | 0 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **7** | **6** | **5** | **4** | **3 2** | **1 0** |
| **0** | **0** | **0** | **1** |  |  |

7 6 5 4 3 2 1 0

2nd.

0

**n (interval)**

1. £ n **.<72**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | ' 1 ’ | **Octave** | 3 | **C** |
| 2 | ’ 2 ’ | **//** | 3 - | **C t** |
| 12 | ’ 12’ | **Octave** | 3 | B |
| 13 | ’ 13' | **//** | 4 | **C** |

|  |  |  |  |
| --- | --- | --- | --- |
| 70 | ’70’ | **Octave 8 -** | -A |
| 71 | ’71 ’ | **- 8** | - A |
| 72 | '72' | **8** | B |

**III**

1st.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **7** | **6** | **5** | **4** | **3 2 10** |
| **0** | **0** | **0** | **1** |  |
| **7** | **6** | **5** | **4** | **3 2 10** |
| **1** | **n** | **(interval)** | | |
|  |  |  | **\*** | **n = same as II** |
| **7** | **6** | **5** | **4** | **3 2 10** |
| **0** | **m** | **(sound duration)** | | |

**spefifies the sound duration.**

2nd.

3rd.

**\* m = same as 1-2, 2nd**

**Specifies the octave**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **7** | **6** | **5** | **4** | **3** | **2 1 0** |
| **0** | **0** | **1** | **0** | **0** | **n (octave)** |

* 0 =£ n ^ 5
* n = 0 Octave 3

1. " 4
2. " 5
3. " 6
4. " 7

**5-8**

|  |  |  |  |
| --- | --- | --- | --- |
| **7** | **6** | **5 4** | **3 2 10** |
| **1** | **0** | A0 | **trans- . | n 'position' |** |
| **0 £ n £ 11** | | |  |
| **> n ^** | **- 0** |  | **7** |
|  | **1** |  | **8** |
|  | **2** |  | **9** |
|  | **3** |  | **10** |
|  | **4** |  | **11** |
|  | **5** |  |  |
|  | **6** |  |  |

1st.

**IV. Specifies the sound volume**

7 6 5 4 3 2 1 0



• 0 £ n £ 15

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **n =15** | **0** | **(dB)** | **9** | **-11 (dB)** | **3** | **-19.5** | **(dB)** |
| **14** | **- 2** | **(dB)** | **8** | **13 (dB)** | **2** | **-21.5** | **(dB)** |
| **13** | **4** | **(dB)** | **7** | **-12.5 (dB)** | **1** | **-23.5** | **(dB)** |
| **12** | **6** | **(dB)** | **6** | **-14.5 (dB)** | **0** | **OFF** | **(dB)** |
| **11** | **7** | **(dB)** | **5** | **16.5 (dB)** |  |  |  |
| **10** | **9** | **(dB)** | **4** | **18.5 (dB)** |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **7** | **6** | **5** | **4** | **3 2** | **1 0** |
| **0** | **1** | **0** | **0** | **n (envelope No.)** | |

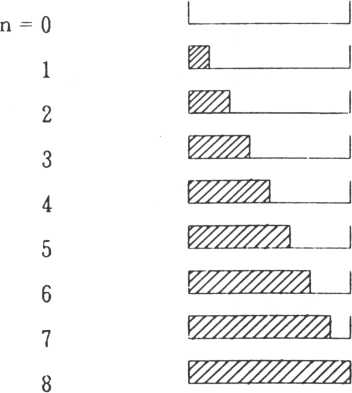
• 0^nS7

**• n** = **0** envelop no. **0** 1 **-** 1 **2-2 3 - 3**

1. **4**
2. **5**
3. 6

7 - 7

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **7** | **6** | **5** | **4** | **3** | **2** | **i** | **0** |
| **0** | **1** | **L°\_** | **1** | **n** | **(hold** | **time no.)** |  |



Sound

duration

: Sounding time period

**VIII**

|  |  |  |  |
| --- | --- | --- | --- |
| **7** | **6** | **5 4** | **3 2 10** |
| **1**° | **1** | **11**° | **j** |
| **7** | **6** | **5 4** | **3 2 10** |
| **0** | **n (sound duration)** | | |

\*: n = same as 1-2, 2nd

**Specifies the tempo**

2nd.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **7** | **6** | **5** | **4** | **3** | **2** | **1 0** |
| **|o** | **1** | **1** | **1** |  | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1 0** |
| **n** | **(tempo)** | | |  |  |  |

• 1 £ n^255

• n :

Quarter notes per minute

7-3 Interrupt by the SML interpreter (user)

When the SML interpreter gets a command from the play buffer in the processing, it can access the user routine.

7-3-1 User routine address storage

Addresses are stored at SEXTA in the numeric information table. At release, an RET instruction stores these addresses.

7-3-2 Transfer parameter

Breg^=3 Channel no. (0,1,2)

Oregon Get pointer

Xreg<£n Relative channel control table head address

Commands executed by the SML interpreter can be got by reading the address obtained by adding the contents of the Xreg and the Creg.

. For a two-byte instruction, the next address data is read also.

For a three-byte instruction, the next address data must be read also.

. Do not fail to reserve Breg. Do not change to interrupt enable.

. Several hundred yseconds should be allowed for the user's processing.

7-4 How to use the noise generator

Tone

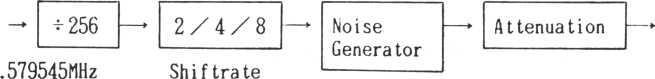
Generator# 3



Clock

N = 3

**Fig. 7-3 Noise generator block diagram**



User-definable items are listed below.

1. Noise source (synchronous, white)
2. Shift rate (N/512, N/1024, N/2048

Tone generator #3)

1. Attenuation (0 dB - OFF)

Chapter 8 Printer Handler

CONTENTS

1. Print-out control by the POUT control flag 3-1
2. POUT instructions 8-2
3. 1 Print-out control by output characteristic flag

8-1-1 Output characteristic flag instruction

76543210

| Outputs LF continuously

following CR (line feed)

Automatic line feed

Tab simulation

Bit 0 — Flag for unconditional output of LF when CR is output as a terminator.

Bit 1 — Flag for automatic output of LF when data reaches the number of logical digits of the printer.

Bit 2 — Flag for tab simulation in eight-column increments when a tab code is received.

Subroutine calling sequence

**8-2 POUT instructions**

Subroutine name: POTCH

|  |  |
| --- | --- |
| Address: | 1799H |
| Function: | Outputs one character |
| Input  Registers | Contents |
| Acc | Output character |

Register save: Acc,B,C,D,E,H,L,X,Y

This subroutine outputs one character to a printer The terminator, output characteristics, etc. are invalid. The head position is updated.

Subroutine calling sequence

Subroutine name: POTLN

|  |  |
| --- | --- |
| Address: | 17C7H |
| Function: | Outputs the text |
| Input  Registers | Contents |
| B | Data size |
| H,L | Data buffer head address |
| Output  Registers | Contents |
| Acc | Data last output |
| B | Data size - bytes of data output |
| H,L | Data address last output + 1 |

Register save: X,Y

This subroutine outputs a text according to the output characteristic flag. Output terminates when the number of bytes of data indicated in register B is output or when a terminator is encountered.

Subroutine calling sequence

Subroutine name: POTBL

|  |  |
| --- | --- |
| Address: | 1805H |
| Function: | Outputs one block of data |
| Input  Registers | Contents |
| B,C | Data size |
| H,L | Data buffer head address |
| Output  Registers | Contents |
| Acc | Data last output |
| H,L | Data address last output + 1 |

Register save: D,E,X,Y

This subroutine outputs the number of bytes indicated in register B. There is no terminator. Control by the output characteristic flag is not provided.

Chapter 9 Other System Information

Contents

1. System table 9-1

CTC interrupt vector

Restart 4-7 interrupt vector Memory range

1. [Application ROM 9-49](#bookmark143)

Save in monitor (header)

Return to monitor

3 . ACMT application program „ . . . 9-52

1. 1 System table explanation (refer to M5 MAP. SR)

9-1-1 System table (26 bytes)

**Address Label (in hexadecimal),**

|  |  |  |  |
| --- | --- | --- | --- |
| 7000 | IVCTCO | **CTC channel 0** | L |
|  |  | **Interrupt vector** | H |
| 7002 | IVCTC1 | **CTC channel 1** | L |
|  |  | **Interrupt vector** | H |
| 7004 | IVCTC2 | **CTC channel 2** | L |
|  |  | **Interrupt vector** | H |
| 7006 | 1VCTC3 | **CTC channel 3** | L |
|  |  | **Interrupt vector** | H |
| 7008 | IVCTC6 | **RST 6** | INST |
|  |  | **Interrupt vector** | L  H |
| 700B | IVCTC7 | **RST 7** | INST |
|  |  | **Interrupt vector** | L  H |
| 700E | SCCDTA | **Control code** | L |
|  |  | **Jump table head address** | H |
| 7010 | SMEMTA | **System memory** | L |
|  |  | **Start address** | H |
| 7012 | SMEMEA | **System memory** | L |
|  |  | **End address** | H |
| 7014 | SUMMTA | **User memory** | L |
|  |  | **Start address** | H |
| 7016 | SUMMEA | **User memory** | L |
|  |  | **End address** | H |
|  |  |  | |

Initial value (in hexadecimal)

|  |  |
| --- | --- |
|  | 6C |
|  | 18 |
|  | 61 |
|  | 18 |
|  | 6C |
|  | 18 |
|  | DF |
|  | 01 |
| k | C3 |
|  | 00 |
|  | 00 |
| k | C3 |
|  | 00 |
|  | 00 |
|  | CD |
|  | 14 |
|  | 00 |
|  | 70 |
|  | 00 |
|  | 80 |
|  | 00 |
|  | 73 |
|  | 00 |
|  | 80 |
|  |  |

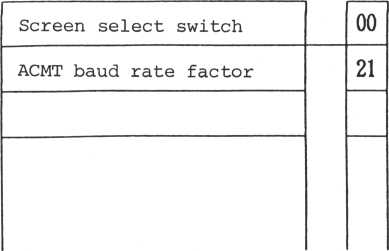
Address label

**7018**

**7019**

SVSSSW

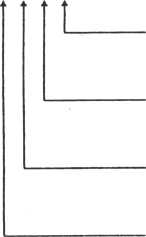
STDLY



SVSSSW flags

76543210

Write page ("0": Page 0, "l": Page 1)



**Layout**

Display screen

("0": Processing screen display, "1": Spare screen display) Display page ("0": Page 0,

"1": Page 1)

("0": Layout I, "1": Layout II)

CTC channel 0 interrupt vector

Jump address when a CTC channel 0 interrupt is requested. A CTC channel 0 interrupt is requested when the SIO signal is received, so CTC channel 0 is usually free.

CTC channel 1 interrupt vector

Jump address when a CTC channel 1 interrupt is requested. CTC channel 1 is used for the music support .

CTC channel 2 interrupt vector

Jump address when a CTC channel 2 interrupt is requested. CTC channel 2 is not used for the system.

CTC channel 3 interrupt vector

Jump address when a CTC channel 3 interrupt is requested. CTC channel 3 is used for screen control.

Control code jump table head address

Head address of a table that provides jump destinations in response to control codes ("00" through "IF") input from the keyboard. When a user defines a control code, the head address of the control code jump table defined by the user must be set here.

System memory start address Head address of RAM memory.

End address of RAM memory.

User memory start address

Head address of the user memory area.

User memory end address

End address of the user memory area.

Screen select switch

Selects page 0 or 1 of VRAM, layout change, etc.

|  |  |
| --- | --- |
| Bit 0 (WRTPGE) | Write page  "0": Selects page 0. "I": Selects page 1. |
| Bit 1 (DSPTBL) | Display screen  "0" : Displays the processing screen.  "1": Displays the spare screen |
| Bit 2 (DSPPGE) | Display page  "0": Displays page 0.  "I”: Displays page 1. |
| Bit 3 (LAYOUT) | VRAM layout "0": Layout I "1": Layout II |
| ACMT baud rate | factor |

The expression for the ACMT baud rate factor is as follows:

1

3 (STn + n2)

**B(BPS)**

**B**-1 **- 84 x 10** 13.4 x 10~6

**B** 1 **- 3nl  
3ST**

B: Baud rate (bps)

T: CPU clock period = 279 (nsec)

n: Number of delay loops

nls Overhead time prior to and posterior

to the loop. (228 ysec)

9-1-2 Keyboard information table (34 bytes)

Address Label (in hexadecimal)

**Initial value  
(in hexadecimal)**

94

|  |  |  |  |
| --- | --- | --- | --- |
| 701A | KINFLG | **Key input control flag** | |
| 701B | KBCTET | **Keyboard conversion table** | L |
|  |  | **Head address of save table** | H |
| 701D | ASWN01 | **Attack switch No. of joypad #1** | |
| 701E | J0YDR1 | **Direction of joypad #1** | |
| 701F | ASWN02 | **Attack switch No. of joypad #2** | |
| 7020 | J0YDR2 | **Direction of joypad #2** | |
| 7021 | JOYPRC | **Interrupt processing of joypad** | L |
|  |  | **Address** | H |
| 7023 | ASWPRC | **Attack switch** | L |
|  |  | **Interrupt address** | H |
| 7025 | RSTPRC | **Reset key** | L |
|  |  | **Interrupt address** | H |
| 7027 | HLTPRC | **Halt key** | L |
|  |  | **Interrupt address** | H |
| 7029 | PHSKAD | **Address of last pressed key** | |
| 702A | LSKYST | **Last pressed auxiliary key** | |
| 702B | LKYADR | **Address of last input-approved key** | |
| 702C | CHATIT | **Chatter prevention counter** | |
| 702D | ARPSTI | **Auto-repeat start time** | I |
| 702E | ARPSTW | **Auto-repeat start time** | W |
| 702F | ARPITI | **Auto-repeat interval** | I |
| 7030 | ARPITW | **Auto-repeat interval** | W |

E7

09

00

00

00

00

2E

00

2E

00

IB

02

IB

02

00

00

00

05

IE

IE

04

04

**\***

**\* \* \***

7031

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  |  |
| KBUFTA | **Keyboard buffer Head address** | L  H |  | DF  70 |
| KDTPPT | **Keyboard buffer put pointer** | |  | 00 |
| KDTGPT | **Keyboard buffer get pointer** | |  | 00 |
| KBFSIZ | **Keyboard buffer size** | |  | 3F |
| KINWTM | **Key input period** | L |  | 00 |
|  |  | H |  | 00 |
| KINWTM | **Key input standard period** | |  | 3C |
| TERMAL | **Terminator from ACELN** | |  | 00 |
| BELKF | **Click frequency factor** | |  | 04 |
| BELKFL | **Click length** | |  | 02 |
|  |  | |  |  |

7033

7034

7035

7036

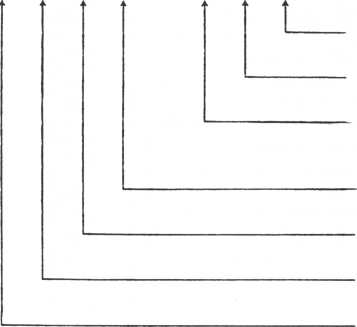
7038

1. 703A 703B

KINFLG flags

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 j 5 | 4 | 3 | 2 | 1 | 0 |

Key input mode (LSB) flag



Key input mode (MSB) flag

Key input mode-change flag ("1" = Already changed)

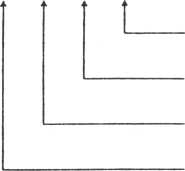
Type ahead mode flag ("1" = Type ahead)

Reset key done flag ("1" = Busy, "0" = Ready) Continuous scrolling flag) ("1" = Continuous scrolling) Click switch flag ("1" = ON, "0" = OFF)

LSKYST flags

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |

Control key flag Function key flag Left shift key flag Right shift key flag Special control key flag



Key input control flags

These flags control information input from the keyboard and joypad.

. Key input mode flag

Indicates keyboard mode.

"00" = Letter mode "01" = Capital mode

"10"= Graphics mode

. Key input mode-change flag

Indicates that the key input mode changed.

. Type ahead mode flag

Switching of pre-press keyboard mode

"0" = Type ahead mode relased »1" = Type ahead mode

. Reset key done flag

"0" = Ready The reset key is enabled.

"1" = Busy The reset key is disabled.

**. Continuous scrolling flag** "0" = No mode change "1" = Continuous scrolling occurred.

. Click switch flag

Switches when a key is pressed. **"0" = OFF**

"1" = ON

. Attack switch No. of joypad

This is not currently supported.

**How the joypad is read** 1

**8 UP 2**



7 LEFT 0 RIGHT 3

**The joypadcis. scanned at every timer interrupt to determine its setting.**

6 DOWN 4

5

. Joypad interrupt address

Control passes to the address when the joypad is moved. If the joypad position does not change, control passes to the preset return instruction address is used.

Attack switch interrupt address

Control passes to this address when the attack switch is pressed. If the attack switch is not pressed, control passes to the preset return instruction address is used.

Reset key interrupt address

Control passes to this address when the reset key is pressed. If the reset key is not pressed, control passes to the preset return instruction address.

Halt key interrupt address

'Control passes to this address when the halt key is pressed. If the halt key is not pressed, control passes to the preset return instruction address.

Address of last pressed key.

Address of last input-approved key

Chatter prevention counter

To prevent chatter when a key is pressed, the signal of a pressed key remains on for a period determined by a counter. After the count reaches 0, key input is enabled.

Auto-repeat start time (I)

. Auto-repeat start time (W)

The number of interrupts possible from the pressing of a key untiL auto-repeat start is set to I. "W" indicates the number of interrupts a counter counts until auto-repeat starts.

. Auto-repeat interval (I)

. Auto-repeat interval (W)

Indicates that the repeat interval interrupts during auto-repeat and is set to I. "W" indicates the work area.

. Keyboard buffer head address

Head address of the area where key input information is stored.

. Keyboard buffer put pointer

The offset from the keyboard buffer head indicates where information is saved in the keyboard buffer.

. Keyboard buffer get pointer

The offset from the keyboard buffer head indicats where input key information is obtained from the key buffer.

Keyboard buffer size

Indicates the size of the keyboard buffer size, which is given by 2n - 1 (n = 1, 2, ....,8).

Key input switching time

Key input period and key input standard period

Indicate the time after key input that another key input can be made. If another key input does not take place within this time, a time-out error occurs. The actual waiting time is given by the following expression:

**Actual wait (sec)**

**key input ( period**

**key input standard period**

**) x 1/60**

If no key input period is specified, a time-out error does not occur (wait time is unlimited). If no key input standard period is specified, it is set to 256 by default.

Terminator from ACELN

ACELN uses two kinds of terminators. They are listed below for reference.

Click frequency factor Fixed.

Click length Fixed.

**9-1-3 Event control table (30 bytes)**

**Address Label (in hexadecimal)**

**Initial value  
(in hexadecimal)**

|  |  |
| --- | --- |
| 703C | EVMGFG |
| 703D | EVMXN0 |
| 703E | EVIFTA |
| 7040 | UEVMGF |
| 7041 | UPCTBI |
| 7042 | UPCTBW |
| 7043 | (JPCNT |
| 7045 | DWCTBI |
| 7046 | DWCTBW |
| 7047 | DWCNT |
| 7049 | CLKBTW |
| 704A | CLOCKS |

**Event management flags**

**Maximum number of events saved**

**Event information table Head address**

**User event management flag**

**Upcount standard time**

**Upcount**

**Downcount standard time**

**Downcount**

**System clock standard time**

**System clock second data**

|  |  |
| --- | --- |
|  | 23 |
|  | 00 |
|  | 00  00 |
|  | 00 |
|  | 3C |
|  | 3C |
|  | 00  00 |
|  | 3C |
|  | 3C |
|  | 00  00 |
|  | 3C |
|  | 00 |
|  |  |

w

w

System clock minute data

|  |  |
| --- | --- |
| **704B** | **CLOCKM** |
| **704C** | **CLOCKH** |
| **704D** | **ALHTM** |
| **704E** | **ALMTH** |
| **704F** | **ALMPRC** |
| **7051** | **EVHPRC** |
| **7053** | **SPRPRC** |
| **7055** | **SPSTUS** |
| **7056** | **SPSTEP** |
| **7057** | **SPSTPC** |
| **7058** | **BELKC** |
| **7059** | **BELC** |

**System clock alarm time (minute)**

|  |  |
| --- | --- |
|  |  |
|  | **00** |
|  | **00** |
|  | **FF** |
|  | **FF** |
|  | **2E**  **00** |
|  | **2E**  **00** |
|  | **2E**  **00** |
|  | **80** |
|  | **02** |
|  | **01** |
|  | **00** |
|  | **00** |
|  |  |

**(hour)**

hour data

Alarm processing Head address

Hourly processing Head address

Sprite interrupt processing Head address

Sprite status

Sprite move step

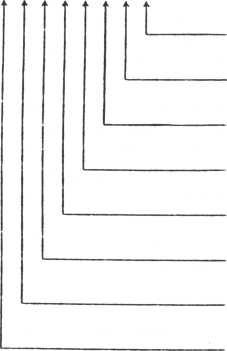
Sprite move step

Keyboard click counter

Bell counter

76543210

Sprite synchronous display ("1" = System clock switch ("1" = ON)



Up counter switch ("1" = ON)

Down counter switch ("1" = pN) Joystick switch ("1" = ON) Keyboard switch ("1" = ON)

User event switch ("1" = ON)

Down counter end flag ("1" = ON)

**ON)**

**UEVMGF flags**

76543210

Event operation flag

("1" = Underevent operation)

SPSTUS flags

76543210

-y -J

I Fifth sprite nurtiber

— Sprite coincidence flag ("1" =

Fifth sprite flag ("1" = True)

Frame interrupt flag (Always "1

**coincidence)**

**")**

These flags are used to manage events.

|  |  |
| --- | --- |
| Bit 0 (SPRSW) | Sprite synchronous display switch Flag to requests support when using an event to move the sprite  "1": Receives support |
| Bit 1 (CLOKSW) | System clock switch  Switch turns the system clock on. |
| Bit 2 (UPCTSW) | Up counter switch  Switch starts the up counter. |
| Bit 3 (DWCTSW) | Down counter switch  Switch starts the down counter. |
| Bit 4 (JOYSW) | Joystick switch  Switch causes the direction of the joystick to be scanned during an interrupt.  "1": ON |
| Bit 5 (KEYSW) | Keyboard switch  Switch causes the keyboard to be scanned during an interrupt. Setting this bit to "0" does not cause the keyboard to be scanned immediately. |
| Bit 6 (UEVSW) | User event switch  Switch causes user events to be  supported.  Setting this bit to "0" stops all user events. |

Bit 7 (DWDNFG) Down counter end flag

When the down counter stops (becomes "0")/ this bit is set to "1".

Maximum number of events saved

Number of user events saved. A maximum of 40 events can be saved.

Event table head address

Head address of the table for saved user events. User event management flag

Prevents other event processing from starting before one user event processing ends. While this flag is "1", it is assumed that a user event is currently operating, and other events cannot be started.

Up counter standard time (I/W)

Up counter

The up counter standard time is the parameter that determines the count of the up counter. "I" indicates the initial value and "W" the work area.

This parameter represents the number of channel 3 interrupts. System interrupts occur every 1/60 second : the count is incremented at this time.

Down counter standard time

Like for the up counter, the down counter standard time indicates the initial setting of a counter, but the count is decremented each time.

System clock standard time

Standard internal time of the computer. The initial value is set to 60, and updates the second counter of the internal clock every second.

Second data of the system clock

Minute data of the system clock

Hour data of the system clock

Indicates the hour, minute, and second of the internal clock.

Alarm time of the system clock (minute)

Alarm time of the system clock (hour)

Head address of alarm processing

At the set time ( hour(s) and minute(s)), control passes to the alarm-processing routine.

Hourly processing head address

Control passes to this address by an interrupt when it is 0 minute and 0 second.

Sprite interrupt processing head address

The head address of the routine for sprite interrupt processing.

Sprite status

Indicates the VDP status register according to the interrupt frame and sets the value.

Bits 0-4 Fifth sprite number

When five sprites are arranged in the same horizontal line, the fifth sprite number is given.

Bit 5 Sprite coincidence flag

When two sprites coincide, this bit is set to "1".

Bit 6 Fifth sprite flag

When five sprites are arranged in the same horizontal line, this bit is set to "l".

Biy 7 Frame interrupt flag

Always "1"

Sprite move step (SPSTEP)

Sprite move step (SPSTPC)

A feature determines how many dots should be moved by one interrupt processing when using a system event to move the sprite. The number of dots (N) is set to SPSTPC, and 2n is set to SPSTEP.

Keyboard click length counter

, Bell counter

Bell tone duration (count)

Event information table (user defined)

|  |  |
| --- | --- |
| **Offset** | **Label** |
| 0 | SUPDLY |
| l | SUPDLH |
| 2 | SUPITV |
| 3 |  |
| 4 | EVPRC |
| 5 |  |

**— Event wait flag**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Support** | | L |
|  |  | **Lag time** | H |
|  | **Support** | | i |
|  | **Interval** | | w |
| **Event .**  **processing ^ Head address** H | | | |

**= wait)**

The event information table is defined by a user for the RAM user area and is used to support user events for the system.

. Support lag time . Event wait flag

When an event is supported, the lag time from support start to processing is set to in is bits of byte 1 and byte 2. Bit 7 of byte 2 is the event wait flag.

If the event is not processed and becomes "0" when the wait flag is "1", the system supports the event.

. Support interval

Determines how many event support intervals and interrupt frames occur for each support processing.

. Event processing head address

Head address of supported event processing.

9-1-4 POUT management table (3 bytes)

76543210

Initial value

Address Label (in hexadecimal)

(in hexadecimal)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 705A | POUTFG | **Output characteristic flag** |  | 07 |
| 705B | PMXCLM | **Maximum number of printers** |  | 50 |
| 705C | PHDPOS | **Virtual head position** |  | 00 |
|  |  |  |  |  |

POUTFG flags

Outputs LF continuously following CR when "1".

Automatic line feed when "1"

TAB simulation when "1"

. Output characteristic flags

Indicate control conditions when text is output from POUT.

Operation when each bit is "1" is shown below.

Bit 0 After CR ("OD") has been output

LF ("OA") is output.

Bit 1 When the head position reaches the

maximum number of digits, CR (or CR and LF) is output automatically.

Bit 2 Provides the tab simulation in increments

of 8 digits. The code output during tab simulation is SP ("20"). (Supports POTLN only.)

Maximum number of digits

The number of logical digits of a device (printer) connected to POUT (80 digits by default).

Head position

The head position of a connected printer. It is set to "0" when CR is output or the head position reaches the maximum number of digits. Otherwise "1" is added each time a character is output.

|  |  |
| --- | --- |
| **Address (in hexadecimal)** | **Label** |
| 705D | SEXTA |
| 705F | SEVPA |
| 7061 | TMPOD |
| 7062 | TMPOC |
| 7063 | MSCAL |
| 7064 | SGSYT |

7072

7080

708E

^ 14 **bytes**

SGSYT2

? 14 **bytes**

SGSYT3

? 14 **bytes**

**User routine Head address**

**Envelope data table Head address**

**Tempo counter constant**

**Tempo counter**

**Transposition data**

**Interpreter**

**Management table 1**

**Interpreter**

**Management table 2**

**Interpreter**

**Management table 3**

**Initial value (in hexadecimal)**

|  |  |
| --- | --- |
|  | 54  18 |
|  | 79  1A |
|  | IF |
|  | IF |
|  | 00 |
|  |  |
|  |  |
|  |  |
|  |  |

. User routine head address

The head address of a user routine which is run whenever a musical performance interrupt is requested. If this is not specified, control passes to the return instruction address.

. Envelope data table head address

The head address of the table of envelope data (up to seven envelopes). Data for one envelope consists of 8 bytes: 4 bytes for rising and 4 bytes for release. (Refer to Appendix 7, the standard envelope table.)

Tempo counter constant

This area is reserved for tempo counter constants, as determined by the tempo instruction.

Tempo counter

Used to set a down counter, which calls the PLAY routine each time the count reaches "0".

Transposition data

Contains a number from 0 to 11 which indicates the basic sounds determined by the transposition instruction.

Example:

|  |  |
| --- | --- |
| **c** | **”0"** |
| **c#** | **111 '■** |
| **D** | **"2"** |
| **Bb** | **(A#)** |
| **B** | **f- "B"** |

Interpreter management table (14 bytes) Offset

(in hexa- Label

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| uyuiinai**;**   1. SGSYTl—3   1  2  3 | **Play buffer Head address** | L  H |  |  |
| **Play buffer size** | | 20 |
| **Put pointer** | | 00 |
| 4 | **Get pointer** | |  | 00 |
| 5 | **Hold time number** | |  | 07 |
| 6 | **Volume** | |  | 00 |
| 7 | **Octave** | |  | 02 |
| 8 | **Default sound length** | |  | 10 |
| 9 | **Envelope data** | L |  | 00 |
| A | **Head address** | H |  | 00 |
| B | **Envelope data get pointer** | |  | 10 |
| C | **Sound length counter** | |  | 00 |
| D | **Hold time counter** | |  | 00 |
|  |  | |  |  |

. Play buffer head address

Indicates the play buffer head address.

. Buffer size

Indicates the number of bytes in the play buffer (254 bytes maximum).

. Put pointer

Indicates the position of the next put operation.

\_ .play buffer . .put Execution address ^head address pointer

Hold time number

Indicates the hold time (0 - 8) .

Volume

Sets the volume of the sound generator (attenuation). Does not work when the envelope is specified.

Octave

Indicates the octave. A range of six octaves, from 3 to 8, can be specified. la (A) of the octave 4 is 440 Hz.

Default sound length

When the sound length is omitted, this value is used.

Envelope data head address

Indicates the head address of the envelope data.

Envelope data get pointer Envelope data get pointer (0 - 15)

Sound length counter

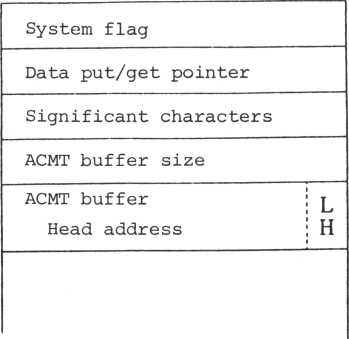
This is a down counter and when it indicates "0", the next command is processed.

Hold time counter

This is a down counter and when it indicates "0", the volume is turned OFF. When the envelope is specidied, moves to the release.

Buffer select switch ("1" = VDP, "0" = CPU RAM) Loading mode ,("1" = Verify)

|  |  |
| --- | --- |
| **Address (in hexadecimal)** | **Label** |
| 708E  708F | SYSFMT  PUTPMT/  GTPMT |
| 7090 | AVILMT |
| 7091 | BSIZMT |
| 7092 | RWBFMT |
| 7094 |  |



**SYSFMT flags**

**76543210**

Write open flag ("1" = True)

. System flags

Used to control saving and loading programs and data to/from ACMT.

Bit 0 BUFSMT Selects the ACMT buffer.

"0": CPU RAM "1": VDP RAM

Bit 1 LDMONT Loading mode

"0": None

"1": Verify

"0": None

"1": Under write open

Data put/get pointer

Used to put or get data to or from the ACMT buffer. Number of significant digits

The number of characters when the data does not fill the buffer completely**,** etc.

ACMT buffer size ACMT buffer size

ACMT buffer head address

ACMT buffer head address

**Address**

**(in hexa- Label decimal)**

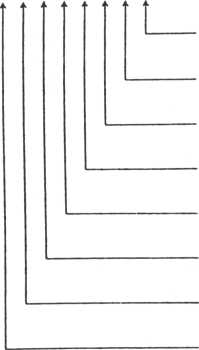
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7094 | DIFLGA | **Screen information flag** | |  | 00 |
| 7095 | CODTLA | **Pattern code table** | L |  | 00 |
|  |  | **Head address** | H |  | 38 |
| 7097 | CCOTLA | **Character color table** | L |  | 80 |
|  |  | **Head address** | H |  | 3B |
| 7099 | CPATLA | **Character pattern table** | L |  | 00 |
|  |  | **Head address** | H |  | 28 |
| 709B | SATTLA | **Sprite attribute table** | L |  | 00 |
|  |  | **Head address** | H |  | 3B |
| 709D | SPATLA | **Sprite pattern table** | L |  | 00 |
|  |  | **Head address** | H |  | 20 |
| 709F | BDCOLA | **Background color** | |  | E0 |
| 70A0 | UPRMVA | **Upper margin of viewport** | |  | 00 |
| 70A1 | LFTMVA | **Left margin of viewport** | |  | 00 |
| 70A2 | HEITVA | **Height of viewport** | |  | 18 |
| 70A3 | WIDTVA | **Width of viewport** | |  | 20 |
| 70A4 | HEITDA | **Height of screen** | |  | 18 |
| 70A5 | WIDTDA | **Width of screen** | |  | 20 |
| 70A6 | CURSYA | **Y coordinate of cursor** | |  | 00 |
| 70A7 | CURSXA | **X coordinate of cursor** | |  | 00 |
| 70A8 | CURADA | **Cursor address in VRAM** | L |  | 00 |
| 70A9 |  |  | H |  | 38 |
| 70AA | CCUCRA | **Character code at cursor position** | |  | 00 |
| 70AB | DISPCA | **Time that cursor appears** | |  | 10 |
|  |  |  | |  |  |

**\* \* \* \***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 70 AC | ERSECA | **Time that cursor does not appear** |  | 10 |
| 70AD | BELFA | **Bell frequency** |  | 08 |
| 70AE | BELFLA | **Bell tone length** |  | 0E |
| 70AF | GRFLA | **Graphics flag** |  | 09 |
| 70BO | GCURYA | **Y coordinate of graphics cursor** |  |  |
| 70B1 | GCURXA | **X coordinate 6f graphics cursor** |  |  |
| 70B2 | GPLPRA | **Dot display processing for graphics II** |  |  |
| 70B4 | GIMPRA | **Image display processing for graphics 2** |  |  |
| 70B6 | MPLPRA | **Pixel display processing for multi-color graphics** |  |  |
|  |  |  |  |  |

**76543210**

Display mode 0



("0" = Overwrite, "1" = Insert)

Display mode 1

("0" = Control execution, "1" = Display)

Screen lockup flag ("0" = Movable, "1" = Locked up)

Cursor status 1

("0": Does not appear, "1": Appears)

Cursor status II

("0": Cursor is in the viewport., "1": Cursor out) VDP display mode M3

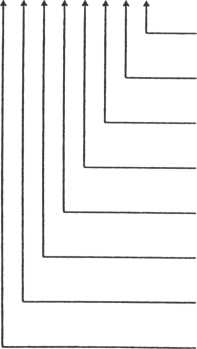
VDP display mode M2

VDP display mode Ml

**GRFLGA flags**

**76543210**

Graphics mode 0 Graphics mode 1 Graphics mode 2 Sprite size 0 Sprite size 1 Sprite size 2 Sprite magnification Sprite solution



Indicates the control status of the processing screen.

Bit 0 DMODEO Display mode 0

"0" = Overwrite mode "1" = Input mode

In the overwrite mode, writes over characters already written - old characters are erased when new characters are written.

In the insert mode, shifts characters already written to the right one position then writes the new character .

Bit 1 DMODEl Display mode 1

"0" = Executes the control "lu = Displays the control

When "0", executes the control code without change. When "1", displays code characters for the control code as reverse characters.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CTRL | + | Character | | corresponds to "0", and | |
| CTRL | + | SHIFT | + | Characters | correspond to "1" |

Bit 2 SCRLOC Screen lockup flag

"0" = Operative "l" = Lockup

Usually trying to display a character at the lower right corner of the screen causes the screen to scroll automatically. In the screen lockup mode,

a character can be entered there without scrolling the screen. The cursor then returns to the home position.

|  |  |
| --- | --- |
| Bit 3 | CURON Cursor status I  n0" = The cursor does not appear "l" = The cursor appears. |

Indicates the cursor blink status.

|  |  |
| --- | --- |
| Bit 4 | CUROUT Cursor status II  "0" = The cursor is in the view\* port.  "1" = The cursor is not in the viewport. |

Indicates whether the cursor is positioned in the viewport or not.

|  |  |
| --- | --- |
| Bit 5 | M3 = VDP display mode |
| Bit 6 | M2 |
| Bit 7 | Ml |

Indicates the screen display mode.

|  |  |
| --- | --- |
| M1M2M3 |  |
| "000" = | Graphics I mode |
| "001" = | Graphics II mode |
| "010" = | Multi-color mode |
| "100" = | Text mode |

Pattern code table head address

Indicates the head address of the pattern name table of the processing screen (address in VRAM).

Character color table head address

Indicates the head address of the character pattern generator.

Sprite attribute table head address

Head address of a table with 32 four-byte records which show the sprite attribute. No meaning in the T mode.

Sprite pattern table head address

Head address of the pattern generator for the sprite.

Background color

The upper four bits indicate character colors in the T mode. The lower four bits indicate the background color (boundary color) in each mode.

Left margin of viewport

Upper margin of viewport

Viewport height

Viewport width

Screen height

. Screen width

Left margin of viewport

x = o

X =31or39

Y= 0

P

**o**

J

**p P C CT> Q) -H <D <U P P O**

4-1

0

**G**

**■H**

**P**

**(U**

**&**

**£**

**Viewport width**

P

**•&**

**•H**

CD

**P**

P

u

o

Q)

**■H**

**>**

**Lower**

**margin**

Right

**Y =23-**

margin

Screen width

(32 or 40)

1 In T mode

The left and upper margins are included in the viewport when the viewport is cut. This means that the X-coordinate of the upper left corner in the viewport is the left margin, and the Y-coordinate is the upper margin.

/Viewport. \_ .lower , \_ .upper , ,

'height 'margin' margin

.Viewport, \_ .right , \_ .left , ,

'width **]** 'margin' 'margin'

The screen width is 40 in the T mode and 32 in other modes. The screen height is always 24.

X and Y coordinates of the cursor

Coordinates on the screen indicate the current cursor position.

Addresses in VRAM of the cursor

The address in the VRAM indicates the current cursor position.

Character code on the cursor position

Saves the character code at the current cursor position.

Cursor appearance time Cursor non-appearance time

Determine the cursor appearance time and non- appearance time when the cursor is blinking.

Bell frequency

Bell tone length

Determine the bell frequency and tone length.

Graphics flags

Used for graphics display.

Graphics modes

Indicate a display procedure for graphics.

Mode 1 Mode 0

0 0 = Replace

**Mode 2**0

0

0

1

Replaces the currently displayed color with the assigned color without changing the assigned color.

1. 1 = OR

ORs the code of the assigned color and the code of the color currently displayed and displays the color for the resulting code.

1. 0 AND

ANDs the code of the assigned color and the code of the color currently displayed and displays the color for the resulting code.

0 0 XOR

XORs the code of the assigned color and the code of the color currently displayed and displays the color for the resulting code.

0 0 = Erase

Erases the assigned color and the currently displayed color so that a new color can be displayed.

Sprite size

Indicates the vertical and horizontal dimensions (in dots) of the sprite.

|  |  |  |  |
| --- | --- | --- | --- |
| Mode 2 | Mode 1 | Mode | 0 |
| 0 | 0 | 1 | = 8 dots x 8 dots |
| 0 | 1 | 0 | = 16 dots x 16 dots |
| 1 | 0 | 0 | = 32 dots x 32 dots |

Sprite magnification

Indicates the magnification used to display the sprite.

"0" = 1 x (no magnification)

"1" = 2 x (double magnification)

(The 8x8 sprite is displayed as 16 x 16 dots.)

Sprite resolution

Indicates the resolution of the sprite.

"0" =8x8 dots "l" = 16 x 16 dots

The following four possibilities are available for the sprite size, magnification, and resolution in relation to the VDP. Sprites are restricted to these four types according to the three sprite flags above.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Resolution | | | Magnification |  | Size | |
| 8 | **X** | 8 | 1 | 8 | **X** | 8 |
| 8 | **X** | 8 | 2 | 16 | **X** | 16 |
| 16 | **X** | 16 | 1 | 16 | **X** | 16 |
| 16 | **X** | 16 | 2 | 32 | **X** | 32 |

. Graphics cursor position (X, Y)

Cursor position in the graphics display mode

In both the GII mode and the multi-color mode, the upper left corner of the viewport is the origin. In the GII mode, the position is indicated by a coordinate system in dot units. In the multi-color mode, the position is indicated by coordinates of a 4 x 4-dot pixel.

\* Graphics handling is not included in the M5 monitor, but it is in the external graphics handler (M5 XGR .SR).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Address (in hexadecimal)** | Spare screen **Label** | table | (36 bytes) | | |
| 70B8 | DIFLGP |  | |  |  |
| ? | ? |  | |  |  |
| 70DA  70DB | MPLPRP |  | |  |  |

The table configuration is same as that for the processing screen information table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Address (in hexadecimal)** | **Label** |  |  |
| 70DC | **FKMGFG** | **Function key management** | **flag** |
| 70DC | FKIFTA | **Function key** | **; L** |
| 70DE |  | **Information head address;** H  **i** | |



FKMGFG flags

**76543210**

I Number of function keys (up to 26)

Function key data format

("0": With a counter,

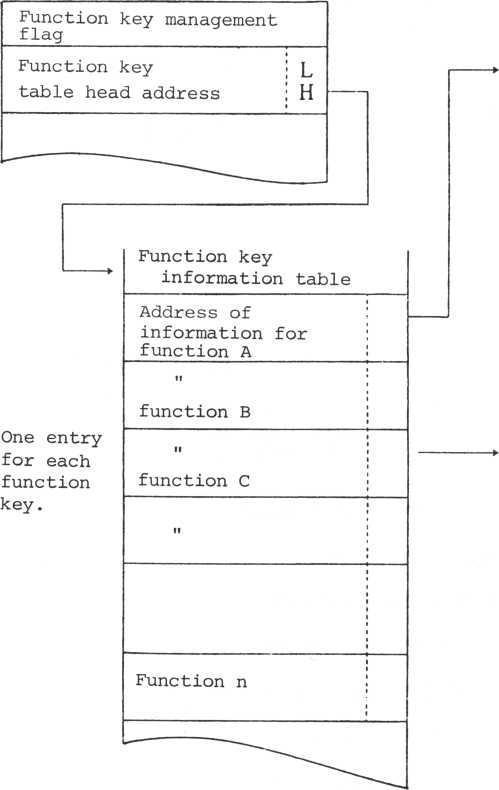
"1": Without a counter)

. Function key management flag Function key data format As data committed to function keys:

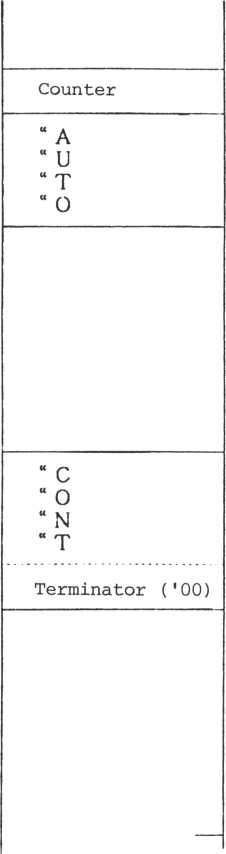
"0" when setting with a counter "1" when setting without a counter Number of function keys

Up to 26 function keys can be assigned.

Function key table head address (See following figure.)



Without a counter With a counter



It is impossible to mix or use data with a counter and data without a counter as existing data.

9-1-9 System buffer

|  |  |  |  |
| --- | --- | --- | --- |
| **Address (in hexadecimal)**  70DF | **Label** |  |  |
| KEYBUF | Keyboard buffer (64 bytes) | |
| 71 IF | ACMTBF | ACMT buffer (64 bytes) | |
| 715F | SGPBF 1 | SG buffer (reg (32 bytes) | 1) |
| 717F | **SGPBF 2** | SG buffer (reg (32 bytes) | 2) |
| 719F | **SGPBF 3** | SG buffer (reg (32 bytes) | 3) |

71BF

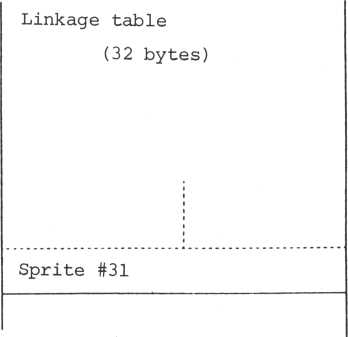
Address (in hexadecimal )

71BF

**Label**

|  |  |  |
| --- | --- | --- |
| MXPSNO | **Maximum number of posts** | |
| SPIFTA | **Sprite information** | L |
|  | **Table head address** | H |
| SPUNK | **Sprite #0** | |
|  | **Sprite #1** | |

Sprite



71C0

71C2

71E1

The post and connecting sprite are not supported by the inter-monitor handler of this computer. Support is implemented by the external sprite handler (M5 XSP .SR).

o Maximum number of posts

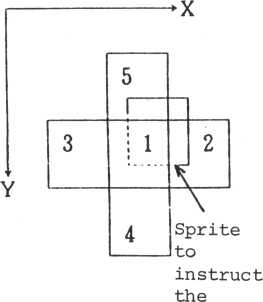
The maximum number of posts saved in the system is 12 when the sprite is supported by the system event. When "0", no support is given.

o Sprite table head address

Head address of the sprite table

o Sprite linkage

Indicates the sprite operating jointly and the connection status. The sprite consists of 32 bytes: #0 through #31.



connection

|  |  |  |
| --- | --- | --- |
| 7 6 5 | 4 3 2 1 0 | |
| t 1 | | [ |

Sprite numbers connected. Connection point:

|  |  |  |
| --- | --- | --- |
| **"000"** |  | **No connection** |
| **"001"** |  | **Overlapping at the same position** |
| **"010"** |  | **Right side (00)** |
| **"011"** |  | **Left side (01)** |
| **"100"** |  | **Lower side (10)** |
| **"101"** |  | **Upper side (11)** |
| **"110"** |  | **f Unused — Do not use.** |
| **"111"** | **:** | J |

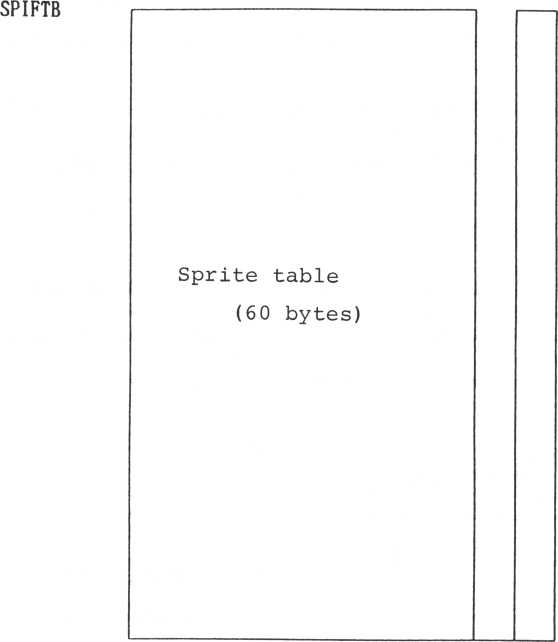
(\*) Refer to sprite connection

handling (extra sprite handler)

Address

(in hexadecimal)

Lable



71E2

721D

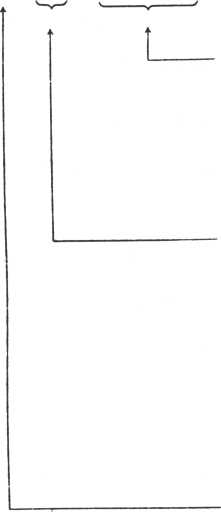
The sprite table is used when the user is supported by the system event and moves the sprite. System support consists of three types: type A, type B, and type C. These types are started by saving a five-byte table called the post. Upto 12 posts can be saved.

**SPIFLG**

6 5

4 3 2 1 0

Sprite numbers



Sprite numbers supported. Sprites connected to other sprites are also supported in the same procedure.

Support format

Identification flag of sprite support format

**"00** " : Type A

**"01** " : Type B

**"10"** : Type C

Support format

Support request flag for posts **"0 " : Does not request support. "1" : Requests support.**

Type A support moves the sprite in a fixed direction. The speed is indicated by the number of interrupt frames per move and has a maximum of 1.

|  |  |  |  |
| --- | --- | --- | --- |
| R | 00 | **Sprite No.** | |
| **Move speed** | | | i  w |
| **Move vector** | | | Y  X |

**Sprite information flag**

The move vector moves by one time support. The number of dots is specified in the (X,Y) direction. When all sprites have disappeared from the screen, support ends.

**. Type B support**

Type B support moves the sprite towards the target position.

The speed is indicated by the number of interrupt frames per move, like type A support.

|  |  |  |  |
| --- | --- | --- | --- |
| R | 01 | **Sprite No.** | |
| **Move speed** | | | i  w |
| **Target position** | | | Y  X |

**Sprite Information flag**

The move each time is determined by the number of sprite move steps in the event management table.

The target position is indicated by a coordinate in dot units for both X and Y. When the sprite reaches the target, support ends.

Type C support moves the sprite relative to the target position. The difference from type B support is that type C support moves to the target position at one time. Hence, the support is completed with one move.

|  |  |  |  |
| --- | --- | --- | --- |
| **R** | **10** | **Sprite No.** | |
| **Objective**  **position Y** | | | **i**  **w** |
| **Objective**  **position X** | | | **Y**  **X** |

**Sprite information flag**

9-2 Application ROM **9-2-1 Save in monitor**

For running the application ROM cartridge under the control of the monitor, a header in the following format must be added to the program head.

Header

|  |  |
| --- | --- |
| **Cartridge identification flag** | |
| **Program auto** | L |
| **Start address** | H |
| **Special IPL** | L |
| **Start address** | H |
| RST4 |  |
| **Jump table** |  |
| RST 5 |  |
| **Jump table** |  |

Applica

tion

program

#1

Applica

tion

program

#2

. Cartridge identification flag

Indicates the capacity and location of the ROM cartridge.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| "00" : | 8K | ROM at | address | 2000H |
| "01" : | 8K | **n** | **ii** | 4000H |
| "02" : | 16K | **ii** | **H** | 2000H |

Program auto-start address

Control passes to this address when the system comes up.

Specify the return instruction address in advance if auto-start is not required, because control passes without exception.

Special IPL start address

During a bootstrapstract, control begins from this address. If this address is not provided, specify the return instruction address in advance.

RST4 jump table

RST5 jump table

When the restart 4 or restart 5 address is used in an application program, control jumps to this address of the header. Thus, you must write an instruction to jump to the restart processing in advance.

If the restarts 4 and 5 are not used in the application program, nothing need be done.

9-2-2 Return to the monitor

To return to the monitor from an application program, control must pass to BTCMT after the following conditions have been established:

1. Initialization of the system table <Routine example>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| LD | HL, | SYTIDI | (System | table initializations) |
| LD | DE, | SYSTBL | (System | table) (Note 1) |
| LD | BC, | SETBLC | (Number | of bytes) |
| LDIR | |  | (11 bytes) | |

1. Set the stack pointer value to the system stack initial value,

SYSTAK = 7300H

1. If interrupt processing is synchronized with music during SML (music routine), reset it.

Restore SEXTA "00"

1. If other special processing is being executed,

reset all of them. Control jumps to BTCMT after the processing has been completed.

Note: Initialize VDP (CAL VDPINT) at the program

head of tape read by BTCMT.

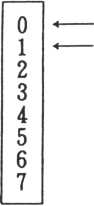
Note 1:

|  |  |
| --- | --- |
|  | M5 |
| SYTDL | "0165"H |
| SYSTBL | "7000"H |
| SETBLC | "005D"H |

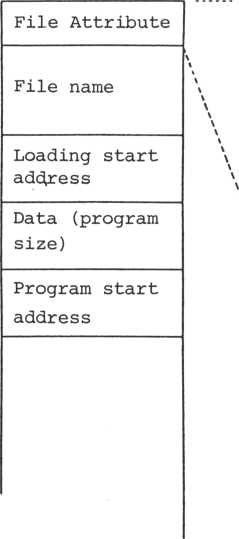
9-3 ACMT application program (called "tape program")

To load an application program from ACMT, execute a TAPE instruction in BASIC and load the program. The program can run as soon as the program load has been completed, but the file attribute in the file ID must be in the execution format.

"0" = Data "1" = Execution format " 0" = Loading only,



**Note:**



"1" = Auto start

This means that the bit 0 of the file attribute is handled as "l". Setting bit 1 to "l" causes immediate execution when the load is completed. When bit 1 is "0", the program is only loaded.

Comment: Loading by chaining several

programs (illustrated below) causes programs A - N to be loaded and automatically start at N.

|  |  |  |  |
| --- | --- | --- | --- |
| **I** | **A** | **I** | **B** |
| **D** |  | **D** |  |

**Loading only**

**•\***

|  |  |
| --- | --- |
| **I** | **N** |
| **D** |  |

L

L

**Loading only**

L

**Auto-start**

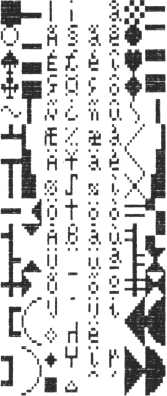
The procedure for loading the next program from tape is same as loading a program from an application ROM.

CONTENTS

1. Standard font table 10-1
2. [10-shift keyboard arrangement 10-2](#bookmark148)
3. [Control code table 10-10](#bookmark190)
4. Error code table 10-11
5. Monitor work table 10-13
6. [**ACMT format 10-14**](#bookmark165)
7. [Standard envelope table 10-17](#bookmark175)
8. [I/O port table 10-19](#bookmark176)
9. Memory map 10-21
10. Color code table 10-24
11. Linkage map 10-25

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | ■”i  jL | T | 45 | hi | ■7  1 |
| Pi |  | ft |  | 0 | Sip | ■% | P |
| i | a | r«i  k^a | 1 | 1 fl Q | | d | q |
| ■“«  **jL** | 13 ft | | II | •“i | BRb | | r |
| **-r** | fSbl | ft # | | “T  .Jl | C S | c | - |
| 4 | !&1 | U | **$■** 4 D T d t | | | | |
| 5 | i\*si  13 | 111 | ■ a. | ET  .J | EU | P | IJ |
| 6 | **\M** | I5i | i\; | hi | F U | ? | I..I |
| -7  r’ | i'rt  i3 | ii |  | -7  f | G lil | '? | I.J |
| **n**  i”i | ! = S  uS j | Si | r | i”i  i”i | HX | h |  |
| 9 | is a | | ) | 9 | I V | i | y |
| fl | y | **rJ**  C3 | + | #  a | JZ | j | “T |
| B | ft | Ki  **Lai** | + | **a**  n | K C k | | /  \ |
| f: | Li | —+ |  |  | L \ | 1 | l’  1 |
| D | **m**  ill | t- | - | — | Ml | Hi | \*1  J |
| E | r | ■ |  | N A | n | ’V |
| F | LI3 |  |  | ■”i | 0 \_ | IJ | 4 |

3 9 H E: C D E F



1. 2 Keyboard arrangement

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **50** | **51** | **52** | **53** | **57** |

RESET

J

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **07** | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **60** | **61** | **62** | **63** | **07** |
|  | | | | | | | | | | | | | |
| **00** | **30** | **31** | **32** | **33** | **34** | **35** | **36** | **37** | **64** | **65** | **66** | **67** | **06** |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **40** | **41** | **42** | **43** | **44** | **45** | **46** | **47** | **54** | **55** | **56** | **03** |

Each key has a code, as shown above. The ten1s digit of

«

the key code indicates the one's digit of the input port for the key. The one's digit of the key code represents the bit position. Input ports have assigned identifiers from "30" to "36".

Ex.: Key "31" corresponds to bit 1 of port "33".

10-2-1 Relationship of keys and character codes in each mode

Letter mode

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **0** | **-** |  | **¥** |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FUNC | q | w | e | r | t | y | U | i | 0 | p | @ | [ | CR |
|  | | | | | | | | | | | | | |
| CTRL | a | s | d | f | g | h | j | k | 1 | » | • | ] | SP |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SHIFT** | **z** | X | **c** | **V** | **b** | n | **m** | > | • | **/** | **\_ SHIFT** |

Auxiliary key information

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | = 1  **i 1** | 0 | 0 | 0 |



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FUNC** | **Q** | **W** | **E** | **R \_** \* | **T** | **Y** | **U** | **I** | **0** | **P** |  | **{** | **CR** |
|  | | | | | | | | | | | | | |
| **CTRL** | **A** | **S** | **D** | **F** | **G** | **H** | **J** | **K** | **L** | **+** | **\*** | **}** | **SP** |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SHIFT | Z | X | C | V | B | N | M | < | > | ? | < SHIFT  i |

**Auxiliary key information**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | **/** | 7 | 0 | 1 1  j O | |

|  |  |
| --- | --- |
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |

**10-2-3 Capital mode**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **0** | **-** | **-** | **¥** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **FUNC** | **Q** | **W** | **E** | **R** | **T** | **Y** | **U** | **I** | **0** | **P @** | | **[** | **CR** |
|  | | | | | | | | | | | | | |
| **CTRL** | **A** | **s** | **D** | **F** | **G** | **H** | **J** | **K** | **L** | » | **•** | **]** | **SP** |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SHIFT** | **Z** | **X** | **C** | **V** | **B** | **N** | **M** | 1 | • | **/** | **\_ SHIFT** |

**Auxiliary key information**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 j 0 | ° | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| I | **))** | **it** | $ | **%** | & | **\*** | ( | ) |  | = |  | i |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FUNC | q | **w** | **e** | r | t | y | u | i | 0 | p | N | { | CR |
|  | | | | | | | | | | | | | |
| CTRL | a | **s** | d | f | g | h | j | k | 1 | + | \* | } | SP |

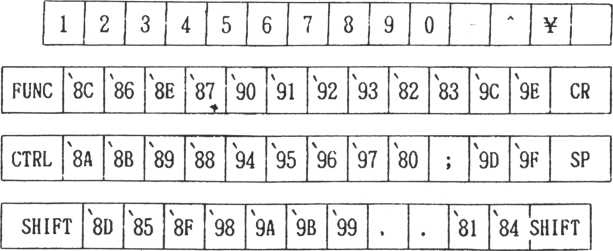
SHIFT

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Z | X | c | V | b | n | m | < | > | 9 | **<** SHIFT  i |

Auxiliary key information

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | **?** | **z** | 1 | 0 |

0 1 1 0 1 I

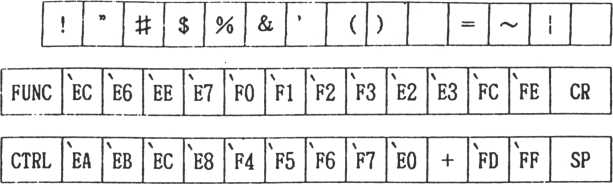


Auxiliary key information

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 j 0 | 0 |

Example:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 5 | 6 | 7 | | 8 | 9 | 0 | | - | \* ¥ |  |
|  | | | | | | | | | | |
| FUNC | a | a | L 1 ■ | 4 | **mmU** | **r.** | g |  | IN | CR |
| CTRL | E | m | 1 | 0 | 1 Ii | | i ii | a |  | SP |
| SHIFT | n | ~j~T~ |  | u |  | | \* | o | ~ SHIFT 1 | |

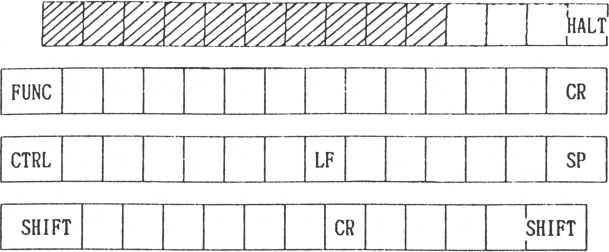


|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | \ | \ | \ |  | \ | \ |  |  | \ | 1 |
| SHIFT | ED | E5 | EF | F8 | F9 | FA | FB | < | > | El | E4 SHIFT  i |

Example:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **[** |  | **' it** | $ **%** & | ( | ) | **0** | **0** | **—]** i  ~ I RESET  i i |
| FUNC | **A** | **01** | - ■  k. x | **aQuonmoi** | | | | |
| CTRL |  | + =1 | **> = 1** | **11** | **□** | **>:DLUQI** | | |
| SHIFT **^** | | **F** |  | **M** | **E** | **0** | O | **l** SHIFT |

Auxiliary key information



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Only when a **key of the tzad section** is pressed.

Special control key

FUNC

Q W

R

**U**

0

CR

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SHIFT |** | **Z** | **X** | **C** | **V** | **B** | **N** | **M** |  |  |  | **l**  **SHIFT**  **i** |

CTRL A

D

H

K

SP

**Auxiliary key information**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 ! °l |  | 0 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Only when —-1 a key of the ^ section is pressed. | L  0 | J  1 |  |  |
|  | 1 | 0 | Effective | only when |
|  | 1 | 1 | bit 1 is | " 0 ". |

1 1

RESET

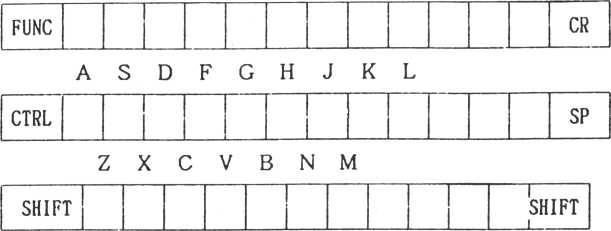
**j** i

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| FUNC | 16 | 22 | 4 | 17 | 19 | 24 | 20 | 8 | 14 | 15 |  |  | CR |
|  | | | | | | | | | | | | | |
| CTRL | i01|S!3 | | | 5 | 6 | 7 | 9 | 10 | 11 |  |  |  | SP |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SHIFT** | 25 | 23 | 2 | 21 | 1 | 13 | 12 |  |  |  | **SHIFT.** |

Eample of function key assignment

QWERTYU I OP



Note: For the details of functions, etc., refer to

the display handler section.

Function

|  |  |
| --- | --- |
| **Code**  **(Hexa**  **decimal)** | **Rela**  **tive**  **key** |
| 00 | @\* |
| 01 | **A** |
| 02 | **B** |
| 03 | C |
| 04 | D |
| 05 | E |
| 06 | F |
| 07 | G |
| 08 | H |
| 09 | I |
| 0A | J |
| 0B | K |
| OC | **L** \* |
| 0D | M |
| 0E | **N** |
| OF | O |
| 10 | P |
| 11 | Q |
| 12 | R |
| 13 | S |
| 14 | T |
| 15 | U |
| 16 | V |
| 17 | W |
| 18 | X |
| 19 | **Y** |
| 1A | **Z** |
| IB | r  **L** |
| 1C | ¥ \* |
| ID | j |
| IE |  |
| IF | **-** |

Jump address routine name

DSPCH3

DSPGM3

SCTOSD

SCRDW

SCRLF

SCRUP

SCRRG

BEL

DELTC

TABLT

LFEED

HOHEP

CLRSC

CRETL

SNTOSD

STOVRM

STINSM

MMODE

GMODE

CMODE

TMODE

NRMSC

REVSC

CRETE

CANCL

RVDSPP

RVWRTP

DSPCH3

RGTAW

LET AW

UPRAW

LWRAW

No operation No operation

Seeks the statement head

Scroll down

Scroll left

Scroll up

Scroll right

Bell

Delete

Tabulation

Line feed

The cursor moves to the upper left corner of the viewport Clear screen

Carriage return (regarded as a line terminator)

Seeks the head of the next statement

Overwrite mode

Insert mode

Multi-color mode

Graphics II mode

Graphics I mode

Text mode

Page 0 is applied to both the display screen and processing screen.

Both the display screen and the processing screen are switched.

Enter (same operation as carriage return) Cancel

Display page switch Processing page switch No operation

Moves the cursor right one character Moves the cursor left one character Moves the cursor up one character Moves the cursor .down one character

Note: \* Keys marked by an asterisk do not correspond

to the keyboard.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Error**  **code** | **Error**  **code**  **(hexa**  **decimal)** | **Comment** | **Sample of processing to return relative error code** | |
| **124** | **7C** | **Spare** |  | **User definition** |
| **125** | **7D** | **Exceeded**  **viewport** | **MVACS**  **RDSCH** | **Cursor move**  **One character read from screen** |
| **126** | **7E** | **Reached**  **target** | **GTSTEP** | **Find a move vector for the sprite to reach the target** |
| **127** | **7F** | **Sprite disappeard from screen** | **MVSPA MVS PR** | **Sprite position change Relative move of sprite** |
| **128** | **80** | **ACMT file is being used** |  |  |
| **129** | **81** | **Checksum**  **error** | **BLOAD**  **RDQPN**  **RFIDC** | **Consecutive area load (ACMT) Read open (ACMT)**  **File ID read** |
| **130** | **82** | **Break key was**  **pressed.** | **BSAVE**  **BLOAD**  **RFIDC** | **Consecutive area save Consecutive area loop File ID read and file name comparison** |
| **131** | **83** | **File name differs.** | **BLOAD**  **RFIDC** | **Consecutive area load File ID read and file name comparison** |
| **132** | **84** | **Data**  **exceeded**  **buffer**  **capacity.** | **RDSTM**  **ETRFK**  **EDTLN**  **ACELN**  **PTKDT** | **One-line read from screen Save of function key Edit input from keyboard 1-text input from keyboard Saves one character from keyboard** |
| **133** | **85** | **Detected EOF/ no data.** | **BLODD**  **GTKDT** | **Data block load**  **Takes one character from**  **keyboard buffer** |
| **134** | **86** | **Display**  **mode**  **inadequate.'** | **STICOL** | **Establishment of character colors** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Error** |  |  |  |
| **Error** | **code** | **Comment** |  | **Sample of processing to return** |
| **code** | **(hexa**  **decimal)** |  | **relative error code** |
|  |
| **135** | **87** | **Time-out** | **ACECH** | **One character input** |
|  |  | **error** | **EDTLN** | **Edit input** |
|  |  |  | **ACELN** | **One line input** |
| **136** | **88** | **Sprite** | **MVS PR** | **Relative move of sprite** |
|  |  | **position**  **exceeded** | **MVS PA** | **Position change of sprite** |
|  |  | **support range.** | **GTSPOS** | **Gets sprite attributed** |
| **137** | **89** | **Inadequate** | **CIEW** | **Viewport establishment** |
|  |  | **command** | **STCHR** | **Character establishment** |
|  |  | **parameter** | **RDCHR** | **Character read out** |
|  |  |  | **STMBF** | **Play buffer set** |
| **138** | **8A** | **Sprite**  **position**  **unknown** |  |  |
| **139** | **8B** | **Underevent** | **STRTDC** | **Down counter start** |
|  |  | **operation/ not continual) le** | **STRTUC** | **Up counter start** |
| **140** | **8C** | **Illegal**  **block**  **detected** | **BLOAD** | **Data block load** |
| **141** | **8D** | **Spare** |  | **User definition** |

Note: Routines to return error codes listed above show

only one example. For further details, refer to the calling sequence of the source list.

It should be noted that many of routines with only one kind of an error do not output an error calling sequence to Acc at the error time, but merely use a carry to decide the error.

Codes not to output MSB ("7C" - "7E") among the error codes are not errors, merely warnings.

7223

7224

7226

7227

724E

7250

7252

7254

7255

UEVCT

UEVPT

LNBUP

LNBP.l

SCRDFR

SCRCOR

CURPSV

SCKLCH

AVRHL

**Event support counter**

**Event support pointer**

**Line buffer**

**40 bytes**

**Data area for scrolling Data area for scrolling**

**Cursor position save area**

ACMT **Work for handler (3 bytes)**

**52-byte work area for LFDTBT LCPAT**

**31-byte buffer for ACMT file ID read**

7279

72C0 SYSTKL

7300

SYSTAK

Unused area

**System stack**

**User area**

**64 bytes**

10-6 ACMT format

. File format

**Add to non- consecutive data**

**Blank**

**run**

**Tape**

**mark**

**File**

**ID**

**l**

B

G

**Data**

**block**

B

G

**Data**

**block**

1

B

G

**Data**

**block**

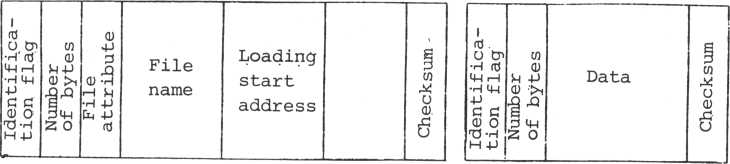
**I**

B

G

**EOF**

**block**



Non-recording time

**Blank run**

**Tape mark: File ID: IBG:**

(about 4 seconds, with a tape recorder having the head-out feature)

8,000 marks (3,000 Hz -\* 2.7 records)

File name, file attribute

Non-consecutive data -\* 3,600 marks Consecutive data 512 marks

**Identification flag:**

|  |  |  |
| --- | --- | --- |
| "H" | (48) : | File ID |
| "D" | (44) : | Data block |
| "E" | (45) : | EOF block |

block is added only when non-consecutive

**Note: The EOF**

data is output.

**File attribute**

1

2

3

4

5

6

7

8

9

1. 11

12

13

**File name**

**(9 characters)**

**Loading start Address**

**Data (program) Size (bytes)**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **0** | **1** |
|  | **0** | **Data** | **Execution format (machine language)** |
| **\** | **1** | **Loading only** | **Auto-start** |
| **\** | **2** | **Conseutive**  **data** | **Data file** |
| **\** | **3** | **CPU memory** | **VDP memory** |
| **\** |  | **Without** | **With** |
| **\** | **4** | **expansion** | **expansion** |
| **\** |  | **attribute** | **attribute** |
| **\** | **5** | **cation ( Z)** |  |
| **\**  **\**  **\** | **6**  **7** | **(2)** |  |

**Language classifi-= 0 cation**

14

15

16

17

18

19

1. 21 22

23

24

25

26

27

28

29

30

**Program Start address**

**Expansion**

**attribute**

**Blank area (13 bytes)**

1. Basic-1 source
2. Falc
3. Basic-G
4. Basic-F
5. Reserved
6. Reserved
7. Reserved

Byte format

(Name) Start bit Data bit Stop bit Total

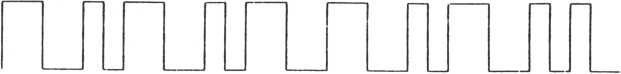
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (Number of bits) |  | (Period) |  |  |
| 1 | ' 0 | ' 2 T |  |  |
| 8 | ' 0 | ' 2 T or ' | ' 1 1 | . **ip** |
| 1 | ' 0 | ' T |  |  |

10 **bits**

Ex.

b

o



M

MSB stop

**bit**

Start **LSB** bit

. Recording wave form: 1-waveform FSK

("0": f, "1": 2f)

. Transmission rate: Optional (1600 - 3200 bps)

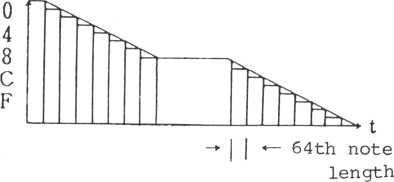
10-7 Standard envelope table

Envelope #0: Without envelope Envelope #1:

Note)

Attenuator

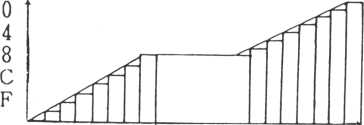
level



**Data: 01234567 89ABCDEF**

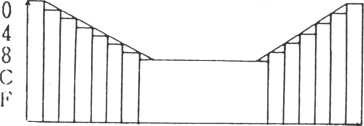
Each step in the envelope is a 64th note at the tempo in force. The duration of the unchanging part of the envelope is found by subtracting the preceding and following envelopes from the total envelope length.

**Envelope #2**



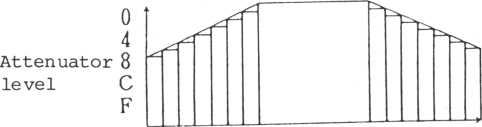
Data: FEDCA98 7654321F

**Envelope #3**



Data: 01234567 7654321F

**Envelope #4**



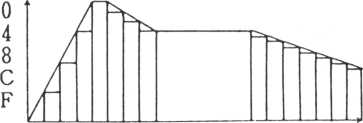
**Data: 76543210 1234567F**

**Envelope 5**



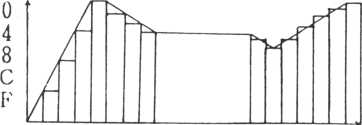
**Data: 02468ACE 2468ACEF**

**Envelope** #6



**Data: FC840123 456789AF**

**Envelope #1**



**Data: FC840123 4543210F**

10-8 I/O port table

|  |  |  |
| --- | --- | --- |
| **Port No. (decimal)** | **Label** | **Description** |
| Z80 C | | T C |
| **00** | **CTCCH0** | **Channel 1 #0: SIO interrupt** |
| **01** | **CTCCH1** | **Channel 1 #1: Peripheral timer** |
| **02** | **CTCCH2** | **Channel 1 #2: SIO clock generator** |
| **03** | **CTCCH3** | **Channel 1 #3: VDP blanking** |
|  |  | **(frame) interrupt** |
| VDP TMS | | 9918A |
| **11** | **VDSTAP** | **Status port** |
| **11** | **VDCNTP** | **Screen base address & control port** |
| **11** | **VDADRP** | **VRAM address port** |
| **10** | **VDRDTP** | **Data read port** |
| **10** | **VDWDTP** | **Data write port** |
| Sound generator SN76489AN | | |
| **20** | **SOUNDP** | **Sound generator control** |
| Keyboard | | |
| **30** | **KEYMD0** | **row #0** |
| **31** | **KEYMD1** | **#1** |
| **32** | **KEYMD2** | **#2** |
| **33** | **KEYMD3** | **#3** |
| **34** | **KEYMD4** | **#4** |
| **35** | **KEYMD5** | **#5** |
| **36** | **KEYMD6** | **#6** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Port No. (decimal)** | | **Label** | **Description** |
| Joystick attack switch | | | |
| **37** | | **JOYDTP** |  |
| Reset/halt key | | | |
| **50** | | **RSTKYP** | **Reset/halt key data port RSTKFG = 7 Data bit position** |
| Cassette tape | | | |
| **50** |  | **ACMTOP** | **Output port**  **ACMTO = 0 data bit position** |
| **50** |  | **ACMTIP** | **Input port**  **ACMTI = 0 data bit position** |
| **50** |  | **M0T0RP** | **Output port**  **MOTOR = 2 ACMT remote switch Bit position = 1** |
| Parallel output | | | |
| **40** |  | **PRTDTP** | **Data** |
| **50** |  | **PRTSBP** | **Strobe**  **STROBE= "0" strobe bit position** |
| **50** |  | **PRTBYP** | **Busy**  **Busy = "1" Busy bit position** |

**Internal map**

’0000

’2000

’4000

’6000 ’7000 ’8000

’ FFFF

-M5.SAV

**(Internal**

**8KB**

**ROM)**

**ROM area**

**External ROM cartridge MAX 16KB**

**Reserved area**

**Internal RAM 4KB**

**External RAM cartridge MAX: 8KB**

**Expansion**

**cabinet**

**MAX: 32KB**

**User**

**RAM area**

Note:

t

**Free area (8KB)**

**Sprite**

**pattern table (Common on page 0 , page 1)**

In case of Layout 1, the addresses 0 through 1FFF of VRAM can be used as a user area.

(2048 bytes)

2800

Character - generator table

(Page 0)

(2048 bytes)

3000

Character generator table (Page 1)

(2048 bytes)

3800

3B00

3B80

3000

3F00

3F80

4000

**Pattern name table (Page 0) (768 bytes)**

**Page 1 Pattern name table (768 bytes)**

**P age 0**

**Sprite attribute table (128 bytes)**

**Page 0**

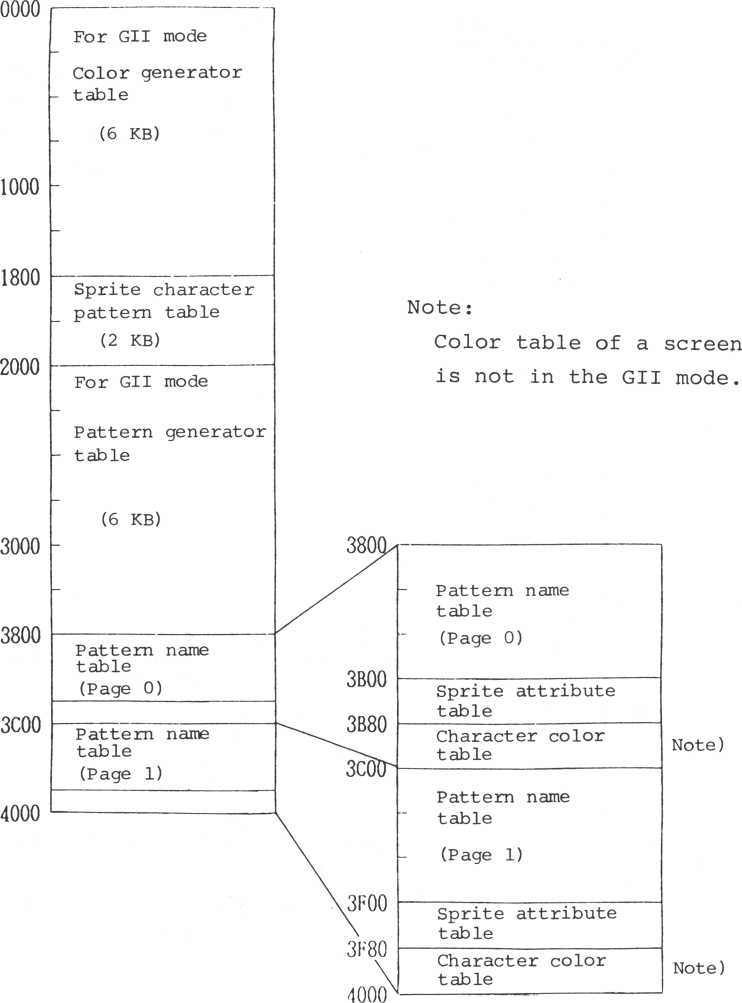
**Character color table (32 bytes)**

**Page 1**

**Sprite attribute table (128 bytes)**

**Page 1**

**Character color table (32 bytes)**



Color code table

10-10

|  |  |
| --- | --- |
| **Color** | **Code** |
| **Transparent** | **0** |
| **Black** | **1** |
| **Green** | **2** |
| **Light green** | **3** |
| **Dark blue** | **4** |
| **Light blue** | **5** |
| **Dark red** | **6** |
| **Cyan** | **7** |
| **Red** | **8** |
| **Light red** | **9** |
| **Dark yellow** | **A** |
| **Light yellow** | **B** |
| **Dark green** | **C** |
| **Magenta** | **D** |
| **Grey** | **E** |
| **White** | **F** |

LOAD MAX = 1FF- C

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| .ACMIU | 003F | .BREAK | 0041 | .BUFFL | 0043 | .CAT RG | 003D |
| .CHKSM | 0040 | .DATOT | 0044 | .EVNRD | 004A | .FINUM | 0042 |
| .ILGBL | 004B | .1LGCM | 0048 | .1LGDM | 0045 | .OVIEW | 003C |
| .SPNFD | 0049 | .SPRDA | 003E | .SPROS | 0047 | .TIMOT | 0046 |
| ACECH | 0827 | ACECH1 | 0845 | ACEDEL | 078A | ACELN | 07A9 |
| ACEST | 066F | ACSCH | OFOO | ADDVCT | 041B | ADJSTD | OOOD |
| ADJSTL | 0018 | ADJSTR | 0023 | ADJSTU | 001E | BEEP | 1181 |
| BEL | 1176 | BELK | 116A | BLKMV | 0B75 | BLKMV2 | 0B81 |
| BLKMVC | ODCB | BLKTRS | 1454 | BLNKC | 0629 | BLOAD | 153B |
| BLOOD | 1579 | BSAVD | 1563 | BSAVE | 1529 | BTCMT | 010D |
| BTCMTO | one | CALKAD | 097B | CALSTV | 047A | CANCL | 13BB |
| CCURML | 148C | CCURMM | 1469 | CCUROT | 1481 | CCUROX | 1486 |
| CCUROY | 1493 | CHGCP | 0707 | CHKRAM | 008E | CHKROM | 00E4 |
| CHKSM | OOCC | CHKSTV | 048E | CHKYM | 073F | CHNGLC | 0D5E |
| CLKBF | 077B | CLOKSP | 0242 | CLRSC | 1393 | CLRSCX | 1394 |
| CLRSS | 137E | CMODE | 0DD8 | CMPCUR | 0784 | CNTLC | 1073 |
| CPATA | 1 ADI | CPATAR | 1AB1 | CPATC | 1B5D | CPATGO | 1C75 |
| CPATG1 | •1EFD | CPATK | 1D75 | CPFNM | 1765 | CRET | 13CD |
| CRETL | 10ED | CTC3EX | 023C | CTC3SP | 01DF | CTCINT | 01C2 |
| CURCOL | 1D25 | CURODX | 1485 | CUROEY | 1492 | CURPTB | 09B7 |
| CVLOF | 118B | CVTIR | 0E61 | DECAD | 08DA | DECFN | 0933 |
| DECTR | 090B | DELSPR | 03C5 | DELTC | OFAB | DIVIDS | 0151 |
| DIVS5 | 0152 | DSPCH | 1088 | DSPCHA | 1083 | DSPCHB | 1082 |
| DSPCHK | 1085 | DSPCU | 064E | DSPLN | 1063 | DSPLTA | 105C |
| DSPLTB | 105B | DSPLTK | 105E | DSPMX | •11ED | DSPSC | 0C64 |
| DWCTSP | 0275 | EDTLN | 0668 | EDTST | 0689 | ERR124 | 003B |
| ERRET | 004 D | ERRTBL | 003B | ERSCU | 0638 | ERSPRA | 1387 |
| ERSSPR | 03CC | ETREIT | 0356 | EXTBL | 1076 | FILSP | 0F27 |
| FNDMY | 049B | FNTRST | 0B5E | FRMSC | 0C77 | GBV | OODF |
| GBVRAM | 14C5 | GBVRID | 0018 | GCURSA | 13F7 | GDIFLG | OOCF |
| GFACMX | 1260 | GFCCP | 0E90 | GMODE | 0B1F | GTAST | 1A4A |
| GTCAPC | 0085 | GTEVMF | 0351 | GTKDT | 08AC | GTSATA | 0369 |
| GTSCLK | 02DB | GTSPLC | 03EE | GTSPOS | 042B | GTSPPC | 0377 |
| GTSTEP | 049F | GTVDMD | 0D34 | HOMEP | 13B6 | IGNORB | 021B |
| IGNORI | 186C | IGNORJ | 186D | INST C | 109F | ITG2M | ODFB |
| JMPHL | 05FE | JOYDTB | 09A7 | JOYSP | 05A5 | KAALPH | 09F3 |
| KAALPS | 0A25 | KAGRPH | OABB | KAGRPS | OAED | KAKANA | 0A57 |
| KAKANS | 0A89 | KBCAT | 09E 7 | KEYSP | 0511 | LCPAT | 0D89 |
| LCPATB | 0B86 | LCPATG | 0B64 | LCPTGC | 0B67 | LDBYT | 16B3 |
| LDBYT1 | 16B6 | LFEED | •10F0 | LFTAW | 10FF | LKUKT | 08F5 |
| LOADC | 1652 | LTBFC | 1556 | LWRAW | 10F6 | M5B0T | 0058 |
| MAGFY | 045C | MLTAL | 1441 | MMODE | 0C44 | MPLAY | 1861 |
| MTROF | 177E | MTRON | 1776 | MULTHD | 142C | MVACS | 13DD |
| MVCURE | 1 IBB | MVSNXL | 0672 | MVSPA | 03CE | MVSPR | 040B |
| NRMSC | 0C5C | NULPAD | 0860 | OUTPS | 163 A | OVRWC | 10CA |
| PAD | 0861 | PADVRM | 0E01 | PAGEM | 0C28 | PAGET | 0C1F |
| PBV | 00D7 | PBVRAM | 14BD | PBVRID | 0010 | PCLMX | 1784 |
| PLAY | 186F | POTBL | 1805 | POTCH | 1799 | POTLF | 1794 |
| POTLN | 17C7 | POTNL | 178D | PTINT | 0C15 | PTINTD | 0C14 |
| PTKDT | 0869 | RATBL | 1650 | RATBLF | 164D | RBTCMT | 0033 |
| RCCSM | OOCA | RCCSMM | 00C7 | RDCHR | 0E75 | RDCNT | 1739 |
| RDFID | 1598 | RDSCH | 14B1 | RDSCHA | 14AC | RDSMX | 120E |
| RDSTM | OEEC | RDVPM | 144B | REVSC | 0E2E | RFIDC | 1587 |
| RGTAW | 10FC | RSTO | 0000 | RST 1 | 0008 | RST2 | 0010 |
| RST3 | 0018 | RST 4 | 0020 | RST5 | 0028 | RST6 | 0030 |
| RST7 | 0038 | RSTRDC | 0336 | RSTRUC | 030D | RSTSP | 05FF |

|  |  |  |  |
| --- | --- | --- | --- |
| RVDSPP | 0C62 | RVWRTP | 0E31 |
| SCRDW | 12BF | SCRLF | 130 A |
| SCTOS | 119E | SCTOSD | 13D5 |
| SGINTS | 181E | S1FTD | 103C |
| SNTOS | 11CA | SNTOSD | 13DA |
| STBCOL | 0C97 | STBYT | 1626 |
| ST ClBL | 0DF8 | STDM1 | 04PF |
| STFRQ | 1A 01 | STFRQD | 1A24 |
| STOPDC | 0340 | STOPUC | 0313 |
| STPCU | 1100 | STPCUD | 1126 |
| STPCUU | 1147 | STRTDC | 0319 |
| STSCHR | 0E58 | STSCLK | 02CF |
| STVFC | 1A3C | STVOL | 1A3A |
| SYTIDT | 0165 | TABLAT | 115A |
| UEVSP | 028F | UPACP | OBEC |
| UPPTP | 08A3 | UPRAW | 10F9 |
| VIEWP | 135C | V1EWRS | 1353 |
| VREGRI | 0471 | WAITST | 1717 |
| WDVLB | 145C | WDVPM | 1460 |
| WTKDTC | 0756 |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| SCCDT | 14 CD | SCNKB | 0966 |
| SCRRG | 12F9 | SCRUP | 12B3 |
| SETBLC | 005D | SGINT | 1811 |
| S1FTL | 0FD3 | SIFTR | 0F3A |
| SRLAD | 16D1 | STOVOL | 1A38 |
| STCCHR | 0E55 | STCHR | 0E59 |
| STEVMF | 0348 | STFCOL | 0C83 |
| STICOL | 0ED3 | STINSM | OF IE |
| STORC | 15ED | STOVRM | 0F19 |
| STPCUL | 1134 | STPCUR | 1116 |
| STRTKT | 0992 | STRTUC | 02FB |
| STSCOD | 0454 | STSCOL | 0445 |
| STVRAO | 14A1 | STVWAD | 149F |
| TABLT | 1158 | TMODE | 0D04 |
| UPCLOK | 02E4 | UPCTSP | 0265 |
| VCTIR | 0E7D | VDPIN r | OEOB |
| VOFAC | 1A31 | VREGI | 0CA3 |
| WATBL | 15EB | WATBLF | 15E8 |
| WRTSC | 0E33 | WTFID | 15C3 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RSTO | 0000 | RST1 | 0008 | ADJ STD | OOOD | PBVRID | 0010 |
| RST2 | 0010 | ADJSTL | 0016 | GBVRID | 0018 | RST3 | 0018 |
| ADJSTU | 00 IE | RST 4 | 0020 | ADJSTR | 0023 | RST5 | 0028 |
| RST6 | 0030 | RBTCMT | 0033 | RST7 | 0038 | ERR124 | 003B |
| ERRTBL | 003B | .OVIEW | 003C | . CATRG | 003D | .SPRDA | 003E |
| . ACfil U | 003F | .CHKSM | 0040 | .BREAK | 0041 | .FINUM | 0042 |
| . BUFFL | 0043 | .DATOT | 0044 | .ILGDM | 0045 | .TIMOT | 0046 |
| .SPROS | 0047 | .ILGCM | 0048 | . SPNFD | 0049 | .EVNRD | 004A |
| .ILGBL | 004B | ERRET | 004D | M5B0T | 0058 | SETBLC | 005D |
| GTCAPC | 0085 | CHKRAM | 008E | RCCSMM | OOC 7 | RCCSM | OOCA |
| CHKSM | OOCC | GDIFLG | OOCF | PBV | 00D7 | GBV | OODF |
| CHKROM | 00E4 | BTCMT | 010D | BTCMTO | one | DIVIDS | 0151 |
| DIVS5 | 0152 | SYTIDT | 0165 | CTCINT | 01C2 | CTC3SP | 01DF |
| IGNORE | 02 IB | CTC3EX | 023C | CLOKSP | 0242 | UPCTSP | 0265 |
| DWCTSP | 0275 | UEVSP | 028F | STSCLK | 02CF | GTSCLK | 02DB |
| UPCLOK | 02E4 | STRTUC | 02FB | RSTRUC | 030D | STOPUC | 0313 |
| STRTDC | 0319 | RSTRDC | 0336 | STOPDC | 0340 | STEVMF | 0348 |
| GTEVMF | 0351 | ETREIT | 0356 | GTSATA | 0369 | GTSPPC | 0377 |
| DELSPR | 03C5 | ERSSPR | 03CC | MVSPA | 03CE | GTSPLC | 03EE |
| MVSPR | 040B | ADDVCT | 04 IB | GTSPOS | 042B | STSCOL | 0445 |
| STSCOD | 0454 | MAGFY | 045C | VREGRI | 0471 | CALSTV | 047A |
| CHKSTV | 048E | FNDMY | 049B | GTSTEP | 049F | ST DM1 | 04FF |
| KEVSP | 0511 | JOYSP | 05A5 | JMPHL | 05 FE | RSTSP | 05FF |
| BLNKC | 0629 | ERSCU | 0638 | DSPCU | 064E | EDTLN | 0668 |
| ACEST | 066F | MVSNXL | 0672 | EDTST | 0689 | CHGCP | 0707 |
| CHKYM | 073F | WTKDTC | 0756 | CLKBF | 0 77B | CMPCUR | 0784 |
| ACEDEL | 078A | ACELN | 07A9 | ACECH | 0827 | ACECHI | 0845 |
| NULPAD | 0860 | PAD | 0861 | PTKDT | 0869 | UPPTP | 08A3 |
| GTKDT | 08AC | DECAD | 08 DA | LKUKT | 08 F 5 | DECTR | 090B |
| DECFN | 0933 | SCNKB | 0966 | CALKAD | 097B | STRTKT | 0992 |
| JOYDTB | 09A7 | CURPTB | 09B7 | KBCAT | 09E7 | KAALPH | 09F3 |
| KAALPS | 0A25 | KAKANA | 0A57 | KAKANS | 0A89 | KAGRPH | OABB |
| KAGRPS | OAED | GMODE | 0B1F | FNTRST | 0B5E | LCPAT G | 0B64 |
| LCPTGC | 0B67 | BLKMV | 0B75 | BLKMV2 | 0B81 | LCPATB | 0B86 |
| UPACP | OBEC | PT1NTD | 0C14 | PTINT | 0C15 | PAGET | 0C1F |
| PAGEM | 0C28 | MMODE | 0C44 | NRMSC | 0C5C | RVDSPP | 0C62 |
| DSPSC | 0C64 | FRMSC | 0C77 | STFCOL | 0C83 | STBCOL | 0C97 |
| VREGI | 0CA3 | TMODE | 0D04 | GTVDMD | 0D34 | CHNGLC | 0D5E |
| LCPAT | 0D89 | BLKMVC | ODCB | CMODE | 0DD8 | STCTBL | 0DF8 |
| ITG2M | ODFB | PADVRM | 0E01 | VDPIN r | OEOB | REVSC | 0E2E |
| RVWRTP | 0E31 | WRTSC | 0E33 | STCCHR | 0E55 | STSCHR | 0E58 |
| STCHR | 0E59 | CVTIR | 0E61 | RDCHR | 0E75 | VCTIR | 0E7D |
| GFCCP | 0E90 | STICOL | 0ED3 | RDSTM | OEEC | ACSCH | OFOO |
| STOVRM | OF 19 | STINSM | 0F1E | FILSP | 0F27 | SIFTR | 0F3A |
| DELTC | OFAB | SIFTL | 0FD3 | SIFTD | 103C | DSPLTB | 105B |
| DSPLTA | 105C | DSPLTK | 105E | DSPLN | 1063 | CNTLC | 1073 |
| EXTBL | 1076 | DSPCHB | 1082 | DSPCHA | 1083 | DSPCHK | 1085 |
| DSPCH | 1088 | INST C | 109F | OVRWC | •10CA | CRETL | 10ED |
| LFEED | 10F0 | LWRAW | 10F6 | UPRAW | 10F9 | RGTAW | 10FC |
| LFTAW | •10FF | STPCU | 1100 | STPCUR | 1116 | STPCUD | 1126 |
| STPCUL | 1134 | STPCUU | 1147 | TABLT | 1158 | TABLAT | 115A |
| BELK | 116A | BEL | 1176 | BEEP | 1181 | CVLOF | 118B |
| SCTOS | 119E | MVCURE | 1 IBB | SNTOS | 11CA | DSPMX | 11ED |
| RDSMX | 120E | GFACMX | 1260 | SCRUP | 12B3 | SCRDW | 12BF |
| SCRRG | 12F9 | SCRLF | 130A | VIEWRS | 1353 | VIEWP | 135C |
| CLRSS | 137E | ERSPRA | 1387 | CLRSC | 1393 | CLRSCX | 1394 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| HOMEP | 13B6 | CANCL | 13BB | CRET | 13CD | SCTOSD | 13D5 |
| SNTOSD | 13DA | MVACS | 13DD | GCURSA | 13F7 | MULTHD | 142C |
| MLTAL | 1441 | RDVPM | 144B | BLKTRS | 1454 | WDVLB | 145C |
| WDVPM | 1460 | CCURMM | 1469 | CCUROT | 1481 | CURODX | 1485 |
| CCUROX | 1486 | CCURML | 148C | CUROEY | 1492 | CCUROY | 1493 |
| STVWAD | 149F | STVRAD | 14 A1 | RDSCHA | 14AC | RDSCH | 14B1 |
| PBVRAM | 14BD | GBVRAM | 14C5 | SCCDT | 14CD | BSAVE | 1529 |
| BLOAD | 153B | LTBFC | 1556 | BSAVD | 1563 | BLODD | 1579 |
| RFI DC | 1587 | RDFID | 1598 | WTF1D | 15C3 | WATBLF | 15E8 |
| WATBL | 15 E B | STORC | 15ED | STBYT | 1626 | OUTPS | 163 A |
| RATBLF | 164 D | RATBL | 1650 | LOADC | 1652 | LDBYT | 16B3 |
| LDBYT1 | 16B6 | SRLAD | 16 D1 | WAITST | 1717 | RDCNT | 1739 |
| CPFNM | 1765 | MTRON | 1776 | MTROF | 177E | PCLMX | 1784 |
| POTNL | 178D | POTLF | 1794 | POTCH | 1799 | POTLN | 17C7 |
| POTBL | 1805 | SG1NT | 1811 | SGINTS | 181E | MPLAY | 1861 |
| IGNORI | 186C | IGNORJ | I860 | PLAY | 186F | STFRQ | •1A01 |
| STFRQD | 1A24 | VOFAC | 1A31 | STOVOL | 1A38 | STVOL | 1A3A |
| STVFC | 1A3C | GTAST | 1A4A | CPATAR | 1AB1 | CPATA | 1 ADI |
| CPATC  CPATG1 | 1B5D  •1EFD | CPATGO | 1C75 | CURCOL | 1D25 | CPATK | 1D75 |

Copyright © 1983 by SORD COMPUTER CORPORATION.

All rights reserved. Printed in Japan.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of SORD COMPUTER CORPORATION., Japan.

**SORD COMPUTER CORPORATION**

**Kyobashi K-l Bldg. 2F., 7-12 Yaesu 2-Chome.**

**Chuo-ku, Tokyo 104, Japan**

**Telephone: (03) 273-3584**

**Telex: 2224225 SORDIN J**

**Cable Address: SORDCOMPCORP TOKYO**

**644 August 1983 Printed in Japan ®**

**SDE-0027**

