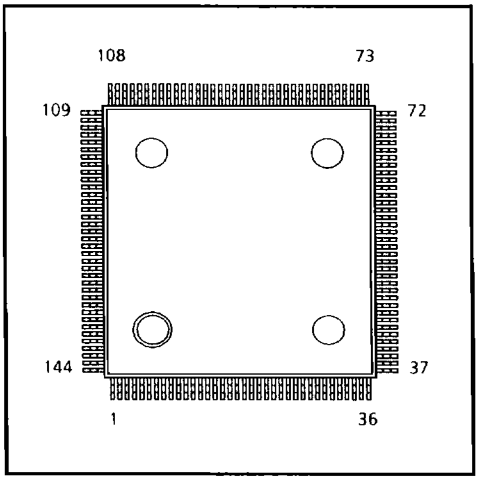
TC8511F

Font Graphics Accelerator

1. GENERAL DESCRIPTION

The Font Graphics Accelerator (FGA) is a high-speed LSI that enables hardware-based processing. Efficient commands are provided to convert display parameters to high-quality characters. Possible output includes Bezier cubic curves, straight lines, arcs (circles), and elliptic arcs (ellipses).



2. FEATURES

- □ Si-gate high speed CMOS technology
- □ +5V single power supply
- □ 144 pin flat package
- □ 13 kinds of basic command
- □ Drawing function
  Bezier curve, Line, Arc (Circle),
  Elliptic arc (Ellipse)
- □ Filling function

Bezier curve dx, dy generation	400 ns/dot	
DDA line dx, dy generation	100 ns/dot	
Outline drawing	400 ns/dot	
Filling	400 ns/byte	(Clock cycle 10MHz)
Command → Character generation (60×60 dot)	1800~2000 character/second	(Estimate value)

- □ Dividing fonts generation
- □ Buffer memory size
  Maximum 32MB (Pair)
- □ Host interface
  32 bits or 16 bits selection
- □ Direct SRAM interface
  DMA transfer function

3. FGA and APPLICATION SYSTEM

3.1 INTERNAL ORGANIZATION

The FGA is composed mainly of the following 5 blocks :

(1) HOST BLOCK

The host block interfaces the host system and the FGA. It provides access to the control registers in the Bezier, DDA and Fill blocks, and to the buffer memory. It has the Host block for the transmission of address displacements between Bezier/DDA block and the host system.

An address displacement (dx, dy) may be generated by the Bezier block, DDA block or host system.

(2) BEZIER BLOCK

The Bezier block generates quantized displacements (dx, dy) to draw an approximated Bezier cubic curve. The approximation is based on 2 on-line points and 2 off-line points defined by the CPU.

The Bezier block creates precise curves that will improve the quality of outline fonts.

Displacements (dx, dy) are generated at an average speed of 400 ns/dot (Clock cycle = 10MHz).

(3) DDA BLOCK

The DDA block generates quantized displacements (dx, dy) to draw approximated straight lines. The approximation is based on 2 on-line points determined by the CPU.

(4) FILL BLOCK

The Fill block has two major functions :

1) Draws in the buffer memory the outlines determined by the address displacements (dx, dy) provided from the Bezier block, DDA block and/or the host system. The buffer memory has a twofold structure; i.e. All outlines are generated in memory A, and outlines indicating special points (points on horizontal lines, vertexes, concaves, overlapped points, corners, etc.) are generated in memory B; and:

2) Fill areas enclosed by the outlines. The result of filling is stored into the buffer memory B.

Operating speed	Outlining :	400 ns/dot
	Filling :	400 ns/byte (Clock cycle = 10MHz clock)

(5) MEMORY BLOCK

The Memory block is connected with the Fill block to interface data exchange to and from the buffer memory (A and/or B).

FGA internal block diagram shown in FIG.3.1.

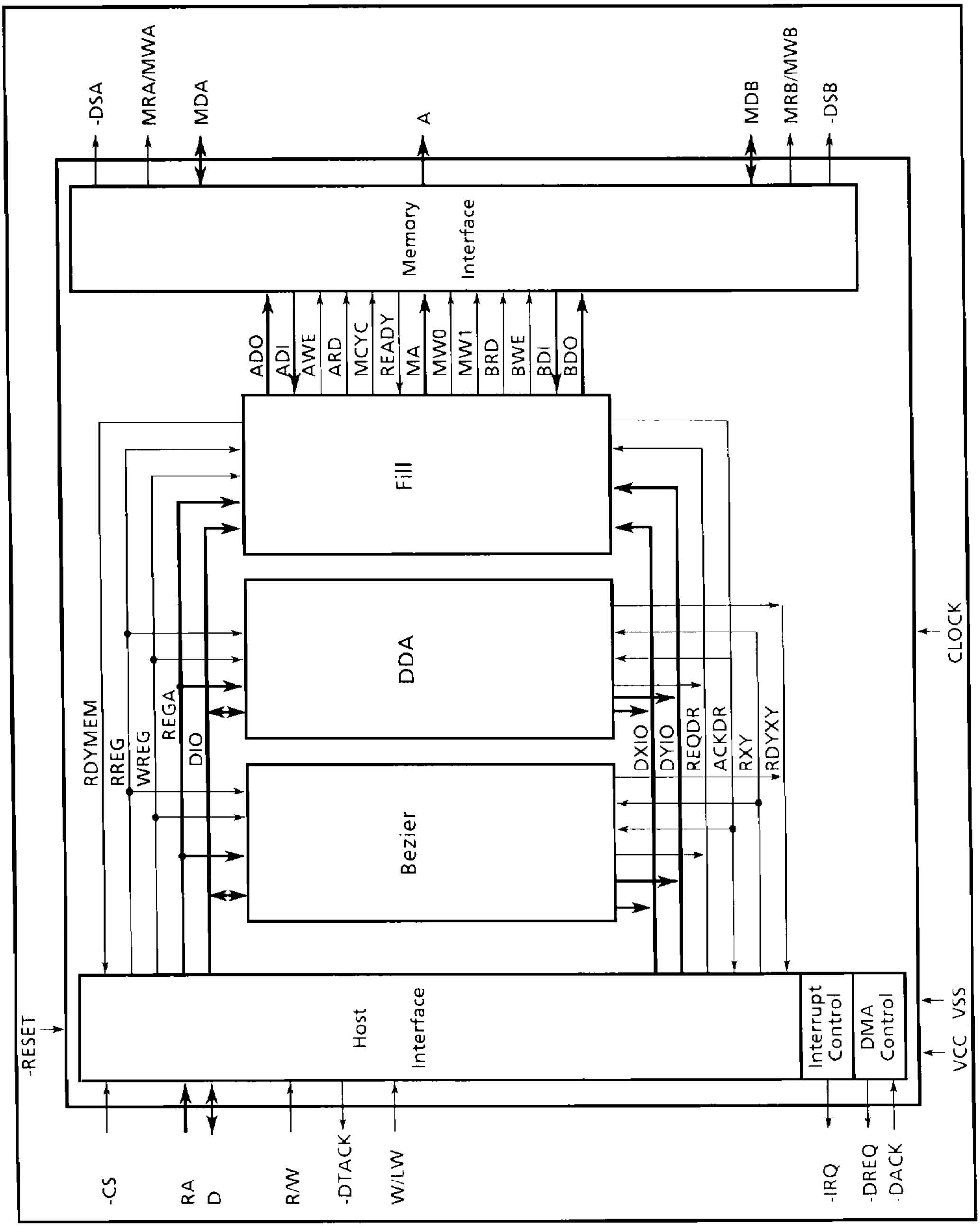


FIG.3.1 INTERNAL BLOCK DIAGRAM

3.2 SYSTEM CONFIGURATION

FIG.3.2 illustrates a typical configuration of the FGA-linked system. The FGA is connected as a peripheral of the host system through the interface.

Commands from the CPU is interpreted by the FGA for converting approximated curves and lines to dot matrix outlines, storing these outlines into dedicated local memories A and B, and filling the outlined areas. To maximize the speed and performance, the buffer memories are separated from the CPU system bus so they operate on independent timing.

When transferring filled data to system memory, it can be done simultaneously with filling motion thus enabling transfer of data to the CPU without lowering system performance.

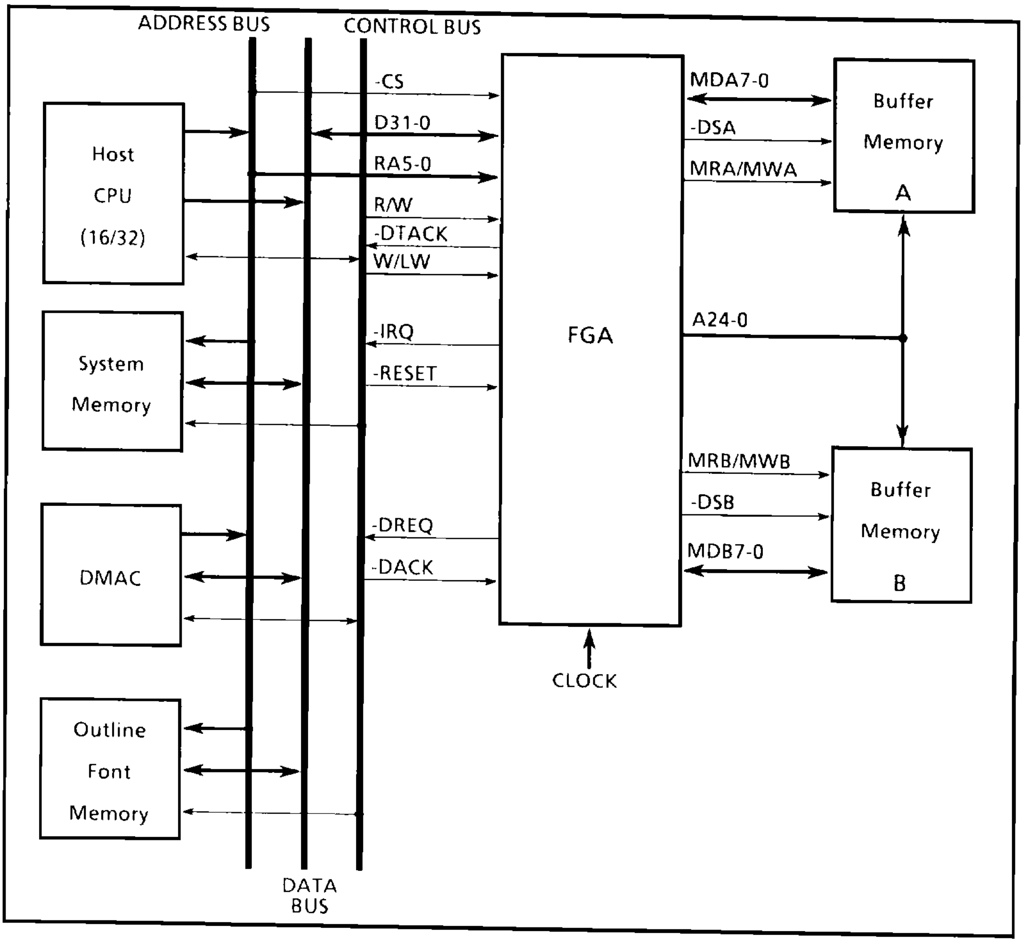


FIG.3.2 EXAMPLE of SYSTEM CONFIGURATION

4. PIN DESCRIPTION

4.1 PIN CONNECTION

NO.	IO	PIN NAME	NO.	IO	PIN NAME	NO.	IO	PIN NAME	NO.	IO	PIN NAME
1	V	VDD	37	V	VDD	73	G	GND	109	V	VDD
2	G	GND	38	G	GND	74	G	GND	110	O	-DSA
3	I	-CS	39	IO	D13	75	V	VDD	111	O	MRA/MWA
4	I	R/W	40	IO	D14	76	O	A0	112	G	GND
5	I	W/LW	41	IO	D15	77	O	A1	113	V	VDD
6	O3	-DTACK	42	V	VDD	78	O	A2	114	IO	MDA0
7	V	VDD	43	G	GND	79	O	A3	115	IO	MDA1
8	G	GND	44	IO	D16	80	O	A4	116	IO	MDA2
9	O	-DREQ	45	IO	D17	81	O	A5	117	IO	MDA3
10	I	-DACK	46	IO	D18	82	G	GND	118	G	GND
11	V	VDD	47	IO	D19	83	V	VDD	119	V	VDD
12	G	GND	48	IO	D20	84	O	A6	120	IO	MDA4
13	OD	-IRQ	49	V	VDD	85	O	A7	121	IO	MDA5
14	I	-RESET	50	G	GND	86	O	A8	122	IO	MDA6
15	V	VDD	51	IO	D21	87	O	A9	123	IO	MDA7
16	G	GND	52	IO	D22	88	O	A10	124	G	GND
17	I	CLOCK	53	IO	D23	89	O	A11	125	V	VDD
18	V	VDD	54	IO	D24	90	G	GND	126	G	GND
19	G	GND	55	IO	D25	91	V	VDD	127	V	VDD
20	IO	D0	56	V	VDD	92	O	A12	128	O	-DSB
21	IO	D1	57	G	GND	93	O	A13	129	O	MRB/MWB
22	IO	D2	58	IO	D26	94	O	A14	130	G	GND
23	IO	D3	59	IO	D27	95	O	A15	131	V	VDD
24	IO	D4	60	IO	D28	96	O	A16	132	IO	MDB0
25	V	VDD	61	IO	D29	97	O	A17	133	IO	MDB1
26	G	GND	62	IO	D30	98	G	GND	134	IO	MDB2
27	IO	D5	63	IO	D31	99	V	VDD	135	IO	MDB3
28	IO	D6	64	V	VDD	100	O	A18	136	G	GND
29	IO	D7	65	G	GND	101	O	A19	137	V	VDD
30	IO	D8	66	I	RA0	102	O	A20	138	IO	MDB4
31	IO	D9	67	I	RA1	103	O	A21	139	IO	MDB5
32	V	VDD	68	I	RA2	104	O	A22	140	IO	MDB6
33	G	GND	69	I	RA3	105	O	A23	141	IO	MDB7
34	IO	D10	70	I	RA4	106	O	A24	142	G	GND
35	IO	D11	71	I	RA5	107	G	GND	143	V	VDD
36	IO	D12	72	V	VDD	108	V	VDD	144	G	GND

4.2 PIN CONFIGURATION

(1) Bidirectional Data Bus (D31-D0: I/O T)

The bidirectional data bus (D31-D0) is used to transfer data between the CPU and the FGA. The FGA uses either a 16-bit or 32-bit bus for data transfer. The data bus provides a 3-state buffer, and is kept at high impedance except for the reading by the CPU. The 16-bit structure should be composed of the bits D15 - D0, with the bits D31-D16 unused.

(2) Read/Write (R/W: Input)

Read/Write (R/W) input signals control the data transfer to and from the CPU. Data are transferred to the CPU when the R/W is high, and to the FGA when the R/W is low.

The data direction is reversed in the DMA transfer.

(3) Chip Select (-CS: Input)

The -CS input signal is used to address the FGA by the CPU. The FGA registers can be read/written only when the -CS signal is low.

(4) Register Address Bus (RA5-RA0: Input)

The RA5-RA0 input signals are used to access the FGA registers (command register, mode registers, parameter registers, status registers and interrupt registers). Each of these register has a identified register number. A register can be read/written only when its register number is designated.

A separate memory access port is automatically selected for the DMA transfer. Therefore, any register number will be ignored if designated.

A register number (address) is composed of one word, but may be accessed by word (16 bits) or long word (32 bits) according to the width of the bidirectional data bus. In the 32-bit structure, reading an even address **n** results in the reading of **n** and **n + 1**, and an odd address **m** results in the reading of **m - 1** and **m**.

(5) Data Transfer Acknowledge (-DTACK: Output T)

The -DTACK signal indicates the end of a data transfer. The signal is based on the 3-state logic.

When the CPU acknowledges a low -DTACK in the read cycle, it terminates the bus cycle after latching data on the bidirectional data bus. In the write cycle, it terminates the bus cycle immediately after it acknowledges a low -DTACK signal. The -DTACK signal is kept at high impedance in non-bus cycles.

(6) Data Bus Width Select (W/LW: Input)

The W/LW signal selects either 32 bits or 16 bits for the bidirectional data bus (D31-D0) width.

The 16-bit structure is selected when W/LW is high; only the D15-D0 lines are used, while D31-D16 are kept at high impedance. When W/LW is low, all lines are used for the 32-bit structure. Normally, the bus width is determined during the system configuration.

(7) Interrupt request (-IRQ: Output OD.)

The -IRQ signal requests the CPU to interrupt the current operation. An interrupt may be requested to terminate a command, divide a line, process an overflow of the stack or check the clipping area.

-IRQ is an open-drain output terminal.

(8) Reset (-RESET: Input)

The -RESET signal resets the FGA state.

When a low -RESET signal is entered, the FGA will:

- Reset the control flip-flop,
- Clear the mode registers,
- Reset a part of the parameter registers and set defaults,
- Clear the interrupt register and the interrupt enable register.

(9) DMA transfer request (-DREQ: Output)

The -DREQ signal is generated to the DMAC (direct memory access control) to request a data transfer in the DMA mode. The system must be set to the DMA mode and ready for a transfer before a -DREQ signal is generated.

Either single transfer or block transfer can be selected.

(10) DMA transfer request acknowledge (-DACK: Input)

The -DACK signal is entered from the DMAC when it acknowledges a DMA transfer request (DREQ). When -DACK is low, data are transferred to the FGA if the R/W signal is high, and to the CPU if R/W is low. While -DACK is low, the -CS signal must be kept at a high level. In a non-DMA mode, -DACK must be pulled up to a high level.

(11) Clock (CLOCK: Input)

The clock signal determines the basic cycle of all the FGA operations.

(12) Memory Data Bus A (MDA7–MDA0: I/O T)

This is a bidirectional data bus used to transfer data to and from the local buffer memory A.

The bus serves as a 3-state buffer. It is kept at a high impedance when the memory is not accessed.

(13) Memory Data Bus B (MDB7–MDB0: I/O T)

This is a bidirectional data bus used to transfer data to and from the local buffer memory B.

The bus serves as a 3-state buffer. It is kept at a high impedance when the memory is not accessed.

(14) Memory Address Bus (A24–A0: Output)

The Memory Address Bus (A24–A0) is used to address the local buffer memories A and B.

(15) Data strobe A (-DSA: Output)

The -DSA signal defines the timing of reading/writing from/to the local buffer memory A.

Data are latched to be read into the FGA from the local buffer memory A at the -DSA rising edge. Data can be written when -DSA is low.

(16) Data strobe B (-DSB: Output)

The -DSB signal defines the timing of reading/writing from/to the local buffer memory B.

Data are latched to be read into the FGA from the local buffer memory B at the -DSB rising edge.

Data can be written when -DSB is low.

(17) Memory Read/Write A (MRA/MWA: Output)

The MRA/MWA signals determine the direction of data to be transferred between the FGA and the local buffer memory A.

Data are read from the local buffer memory A when MRA/MWA is high, and written to the memory when the signal is low.

(18) Memory Read/Write B (MRB/MWB: Output)

The MRB/MWB signals determine the direction of data to be transferred between the FGA and the local buffer memory B.

Data are read from the local buffer memory B when MRB/MWB is high, and written to the memory when the signal is low.

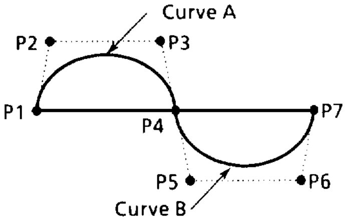
5. FUNCTIONAL DESCRIPTION

5.1 GENERAL

The Font Graphics Accelerator outlines and fills using various parameters, modes and commands provided from the CPU. This section describes the procedures required to achieve specific operations.

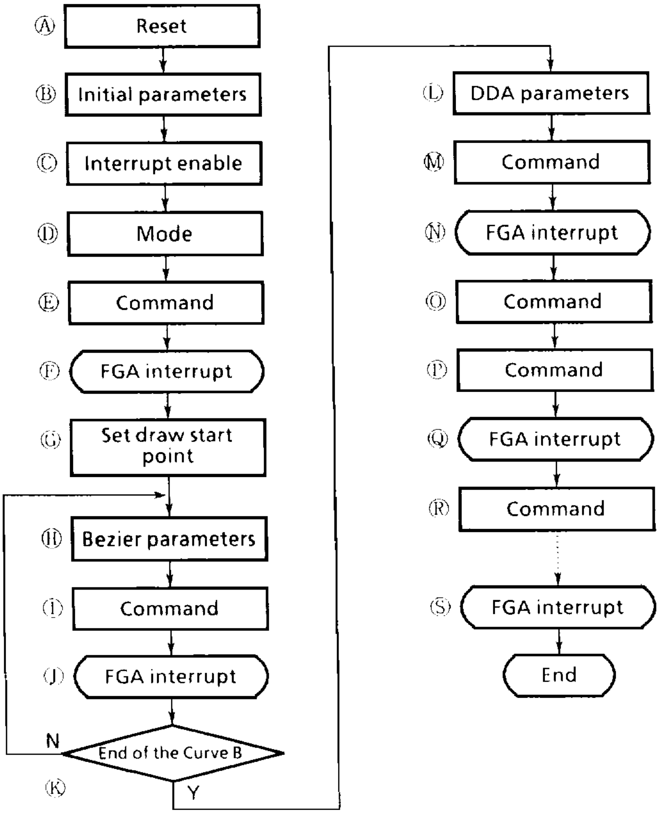
Example

The following pattern can be generated to the CPU as described in the flow chart below.

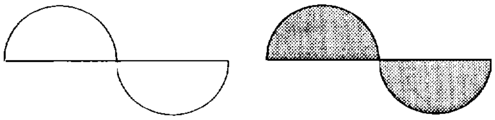


- The curve A inside the points P1, P2, P3 and P4 is drawn by a Bezier command.
- The curve B inside the points P4, P5, P6 and P7 is drawn by a Bezier command.
- The line from the point P7 to the point P1 is drawn by a DDA command.

Process Flow

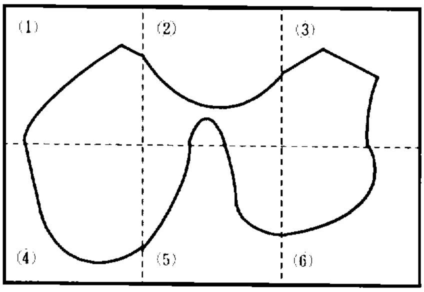


NO.	OPERATION	REGISTER(S) USED
Ⓐ	Initialize the FGA. The FGA is initialized by RESET signal or CRST command.	CMD
Ⓑ	Set the following initial parameters for operating the FGA: · Size of the local buffer memories · Work areas · Start point (P1)	MAD ASD,AED
Ⓒ	Set the interrupt enable bit if an interrupt is required to terminate each command.	IER
Ⓓ	Select a mode for commands to be executed.	MOD1,MOD2
Ⓔ	Execute a CMCL command to clear the local buffer memories A and B.	CMD
Ⓕ	The FGA sets the interrupt bit (INTMB) and generates an interrupt request (-IRQ) to the CPU. The CPU reads the IR register to confirm that the local buffer memories are cleared.	IR
Ⓖ	Set position value of P1 for drawing start point	DS
Ⓗ	Set the coordinates of the points P1, P2, P3 and P4 on the parameter registers (for drawing the Bezier curve A in the local buffer memories).	BX0,BX1, BX2,BX3, BY0,BY1, BY2,BY3
Ⓘ	Execute a CBZ1 command to draw the Bezier curve A.	CMD
Ⓙ	The FGA sets the interrupt bit (INTB1), and generates an interrupt request (IRQ) to the CPU. The CPU reads the IR register to confirm that the Bezier curve A is drawn.	IR
Ⓚ	The CPU checks if the Bezier curve B is drawn. If the curve is not drawn, it repeats the steps 7 through 9 for the points P4, P5, P6 and P7.	
Ⓛ	The CPU calculates and set on the parameter registers the parameters used by the DDA block to draw the line from the point P7 to P1.	DDAR0, DDAR3, DDAR4, DDAC
Ⓜ	Execute a CDDA1 command to draw the straight line.	CMD
Ⓝ	Upon finishing the drawing of straight line, the FGA sets the interrupt bit (INTD) and generates an interrupt request (IRQ) to the CPU which reads the IR register to confirm that the drawing of straight line by DDA is finished.	IR
Ⓞ	Execute a CPTE command to close the outline.	CMD
Ⓟ	Execute a CFIL1 command to fill the closed area.	CMD

NO.	OPERATION	REGISTER(S) USED
Ⓠ	The FGA sets the interrupt bit (INTMB) and generates an interrupt request (IRQ) to the CPU. The CPU reads the IR register to confirm that the filling is finished. At this moment, the following patterns are stored in the buffer memories A and B: Memory A Memory B 	IR
Ⓡ	Execute a CBLK command to transfer to the CPU the C filled pattern stored in the local buffer memory B. The CPU continues reading the results of processing through handshaking communication.	CMD
Ⓢ	When the last word (or long word) is read by the CPU, the FGA sets the interrupt bit (INTMB) and generates an interrupt request (IRQ) to the CPU. After the CPU reads the last word (or long word), it reads the IR register to confirm that the block transfer is finished.	IR

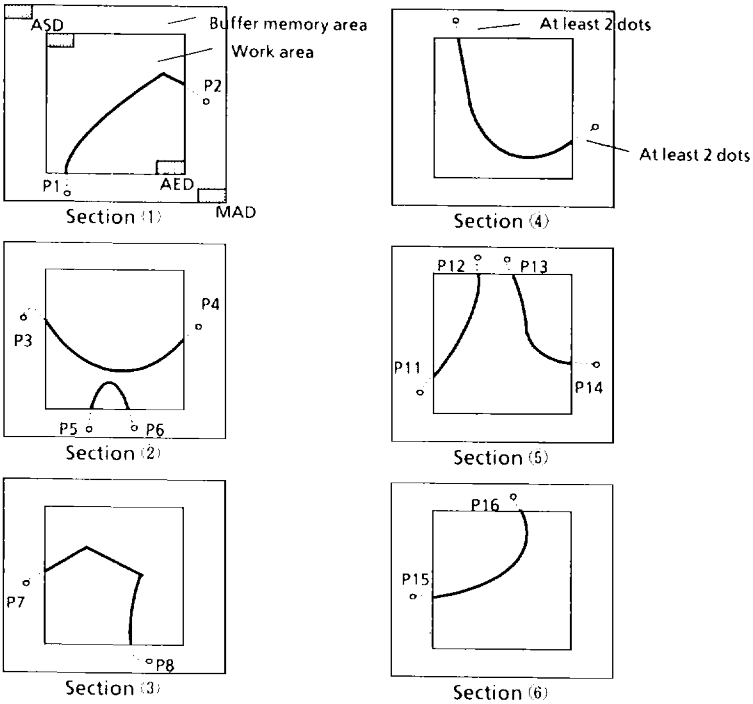
5.2 DIVIDING FONTS

A font larger than the buffer memory area (MAD) can be created in a divided form. Very large fonts can be used because they can be divided into as many sections as desired.



To draw the left pattern, the CPU divides it into 6 sections that fit the size of the buffer memory. The CPU calculates the coordinates of the beginning and end of each outline. Each outline should begin and end at points at least 2 dots extended out of the work area.

The CPU provides parameters of all outlines contained in each divided section.



Section (1)

Section (4)

Section (2)

Section (5)

Section (3)

Section (6)

The work area should be sized smaller than the buffer memory area so that each out line can be extended over its boundary.

A divided pattern can be output to the CPU in the following procedure:
Set parameters → Outline → Fill → Block transfer → Clear the work area.
Divided sections should be processed in the order of ① → ② → ③ → ④ → ⑤ → ⑥. The following fill commands are used to fill the closed area:

CFIL3 command to fill the sections ① and ④
CFIL4 command to fill the sections ② and ⑤
CFIL2 command to fill the sections ③ and ⑥

5.3 DESCRIPTION of the COMMANDS

The FGA supports 13 commands, each of which is assigned to a specific bit in the command register (CMD). A command can be activated by writing "1" to the corresponding bit.

Bit assignment																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
▨	▨	▨														
——			CBLK	CMCL	CFIL4	CFIL3	CFIL2	CFIL1	CPTE	CDDA2	CDDA1	CBZ2	CBZ1	CSTC	CRST	Name of corresponding command.

Commands Assigned to the Command Register Bits

See Section 5.7 for restrictions involved in activating these commands.

5.3.1 RESET COMMAND [CRST]

A CRST command is executed to initialize the FGA. It has the same function as the -RESET signal. Executing a CRST command causes the following operations :

1. Reset the control flip-flop.
2. Clear the mode registers (MOD1 and MOD2).
3. Clear the parameter registers and set defaults.
   - Clear the Work Area Start (ASD) register.
   - Reset the Work Area End (AED) register (to XAE = 63 and YAE = 511).
   - Clear the Current Pointer (CP).
   - Clear the Drawing Start (DS) register.
   - Reset the Buffer Memory Area Definition (MAD) register (to XMA = 6).
   - Clear the Stack Pointer (BSP).
4. Clear the following interrupt registers:
   - Interrupt register (IR).
   - Interrupt enable register (IER).

5.3.2 CLEAR STACK [CSTC]

A CSTC command clears the Bezier Stack Pointer (BSP) used for the generation of Bezier curves. A CSTC command must precede a CBZ2 command that subdivides a Bezier curve into lines.
The stack pointer can be cleared by executing a CSTC command or writing "0".
A CSTC command is not necessary for executing a CBZ1 command (that generates a Bezier curve).

5.3.3 COMMAND to GENERATE a BEZIER CURVE [CBZ1]

A CBZ1 command generates a Bezier cubic curve.
When a CBZ1 command is executed, the Bezier block generates 4-bit quantized displacements (dx, dy) to draw an approximated Bezier cubic curve. A Bezier cubic curve is defined using 4 points, whose x-y coordinates are indicated by the parameter registers BX0–3 and BY0–3.
After a CBZ1 command is executed, the FGA sets the bit INTB1 of the interrupt register.
A CBZ1 command causes one of the following operations according to the mode specified by the MOD1 mode register:

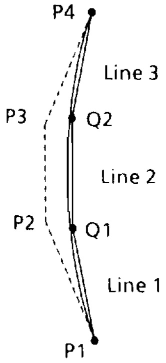
(a) Draw an outline in the local buffer memory A using the displacements (dx, dy).
(b) Draw an outline in the local buffer memory A and the local buffer memory B using the displacements (dx, dy).
(c) Transfer the displacements (dx, dy) to the CPU while doing the operation (a) or (b).
(d) Transfer the displacements (dx, dy) to the CPU without doing the operation (a) or (b).

(a) is selected to draw an outline. (b) is selected to draw an outline and fill. (c) and (d) are used if displacements should be processed by the CPU.

A CSTC command is not required before executing a CBZ1 command.

5.3.4 COMMAND to DIVIDE a BEZIER CURVE [CBZ2]

CBZ2 commands are used to subdivide a Bezier cubic curve into straight lines.
The number of lines is determined by a value set on the flatness register. A Bezier curve to be subdivided is specified by the parameter registers BX0~3 and BY0~3.
The first CBZ2 command determines the first division point (or the end of the first line). To obtain the next division point, another CBZ2 command should be executed.
Each time a division point is determined by a CBZ2 command, the FGA sets the INTB2 bit of the interrupt register, and generates an interrupt request IRQ.



P1–P4	:	Bezier parameters (coordinates of the 4 points)
(P1 and Q1)	:	Beginning and end of the line 1
(Q1 and Q2)	:	Beginning and end of the line 2
(Q2 and P4)	:	Beginning and end of the line 3

The coordinates of the beginning and end of each line are stored in the parameter registers (BX0, BY0) and (BX3, BY3), and read by the CPU.
When the last division is completed, the FGA sets the interrupt bits INTB1 and INTB2, and generates an interrupt request IRQ to inform the CPU of the end of division.
Only the ends (BX3, BY3) should be picked up each time to read subdivided lines continuously.
A CSTC command must be executed before the first CBZ2 command.
A CBZ2 command defines the beginning and end of a line for reading by the CPU. It can be coincided or paralleled with a CDDA1 command to draw this line in the local buffer memories A and B.

The CPU reads the beginning (BX0, BY0) and end (BX3, BY3) of a line, and executes the next CBZ2 command if necessary. At the same time, it calculates and sets on the parameter registers DDAR0, DDAR3 and DDAR4 the parameters required to draw the line just determined. Then, a CDDA1 command can be executed to draw this line in the buffer memories.

The above steps are repeated until the last line is drawn in the local buffer memories A and B.

5.3.5 COMMAND to GENERATE a STRAIGHT LINE [CDDA1]

A CDDA1 command is used to generate a straight line.

When a CDDA1 is executed, the DDA block generates 4-bit quantized displacements (2 bits each for x and y coordinates), the number of which is designated by the DDAC register. A line is drawn based on these displacements and the parameters provided by the parameter registers DDAR0, DDAR3 and DDAR4. After a CDDA1 command is executed, the FGA sets the INTD bit of the interrupt register. A CDDA1 command causes one of the following operations according to a mode specified by the MOD1 mode register:

(a) Draw an outline in the local buffer memory A using the displacements (dx, dy).
(b) Draw an outline in the local buffer memories A and B using the displacements (dx, dy).
(c) Transfer the displacements (dx, dy) to the CPU while doing the operation (a) or (b).
(d) Transfer the displacements (dx, dy) to the CPU without doing the operations (a) or (b).

(a) is selected to draw an outline pattern. (b) is selected to draw outline and filled patterns. (c) and (d) are used if the displacements need processing by the CPU.

5.3.6 COMMAND to GENERATE a CIRCLE, ARC or ELLIPSE [CDDA2]

CDDA2 commands are used to generate a circle, arc or ellipse.

When a CDDA2 command is executed, the DDA block generates 4-bit quantized displacements (2 bits each for x and y coordinates), the number of which is designated by the DDA counter register (DDAC). A 1/8 circle or ellipse is drawn using these displacements and the values of the parameter registers (DDAR0, DDAR1, DDAR2, DDAR3 and DDAR4) and the mode register MOD2. These parameters are preset to define the beginning of a circle/ellipse and its radius. After a CDDA2 command is executed, the FGA sets the INTD bit of the interrupt register.

The operation of a CDDA2 command is determined by the MOD1 mode register, as in the case of a CDDA1 command.

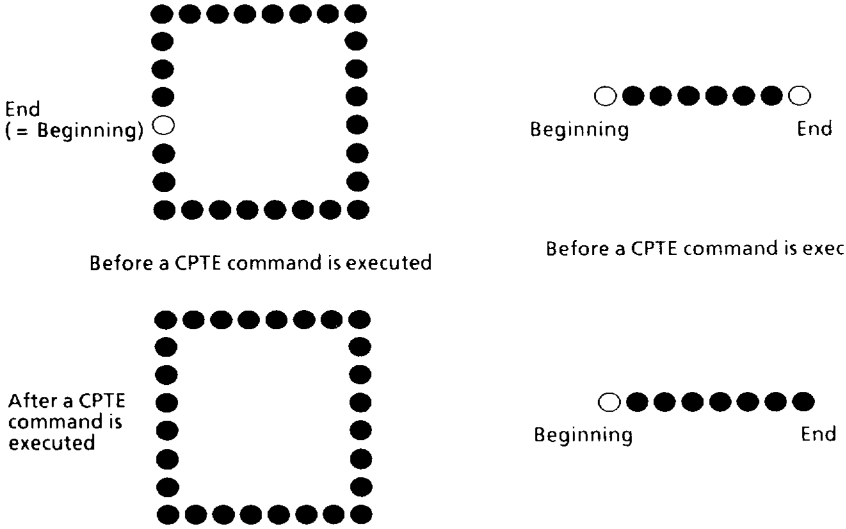
5.3.7 PATH END [CPTE]

A CPTE command is used to close an outline. An outline can be closed only if its end overlaps its beginning. This command should be executed after defining the x – y coordinates of the end (= beginning) of an outline to be closed. To fill a closed area, use the fill commands CFIL 1 to 4.

A CPTE Command example to Close An Outline in the Buffer Memory A

The following outline can be closed by a CPTE command (end = beginning):

The following outline cannot be closed by a CPTE command (end ≠ beginning):



Before a CPTE command is executed

Before a CPTE command is executed

After a CPTE command is executed

After a CPTE command is executed

○ = Blank

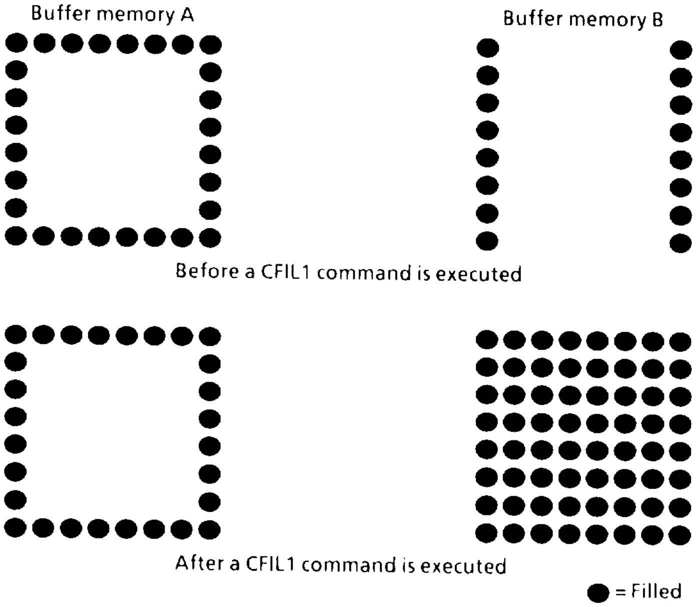
● = Filled

5.3.8 COMMAND to FILL a NON-DIVIDED FONT [CFIL1]

A CFIL1 command is used to fill the interior of a closed outline. Before executing this command, an outline must be drawn by CBZ1, CDDA1 and/or CDDA2 commands or the direct write access of the displacement from the CPU to the dx, dy port, and closed at a point specified by the CPU. A closed area can be filled by a CFIL1 command if it is drawn in both the local buffer memories A and B according to the mode specified by the MOD1 mode register.

After a CFIL1 command is executed, the FGA sets the INTMB bit of the interrupt register.

An Example of CFIL1 Command



Before a CFIL1 command is executed

After a CFIL1 command is executed

● = Filled

A CFIL1 command causes one of the following operations as determined by the MOD1 mode register:

(1) Transfer the filled pattern to the CPU.

(2) Write the filled pattern in the buffer memory B.

(3) Do (1) and (2) simultaneously.

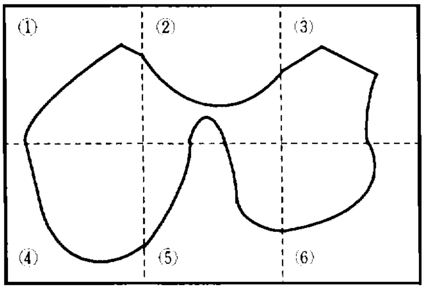
(4) While doing (1), clear the work areas of the buffer memories A and B.

The operation (1) provides the fastest output to the CPU. A block transfer (CBLK command) should be used to send the result of (2) to the CPU. (4) enables outlining and filling to be repeated without using a CMCL (clear memory) command.

5.3.9 COMMANDS to FILL a DIVIDED FONT [CFIL2, CFIL3, CFIL4]

These commands fill a divided font as follows:
CFIL2 command fills the right sections of a divided font.
CFIL3 command fills the left sections of a divided font.
CFIL4 command fills the remaining sections of a divided font.

After a CFIL command is executed, the FGA sets the INTMB bit of the interrupt register.



A font larger than the buffer memory area can be divided into 6 sections as shown in the left figure. To fill this divided font, use the following commands:

CFIL3 command to fill the sections (1) and (4).
CFIL4 command to fill the sections (2) and (5).
CFIL2 command to fill the sections (3) and (6).

These sections must be filled in the order of (1) → (2) → (3) → (4) → (5) → (6).

The operation of a CFIL command 2, 3 or 4 is determined by the MOD1 mode register, as in the case of a CFIL1 command.

5.3.10 CLEAR MEMORY [CMCL]

A CMCL command clears a work area specified by the parameter registers ASD and AED. To clear a work area, set the mode register (MOD1) to specify which memory (A or B) should be cleared, and if it should be cleared to "0" (blank) or "1" (filled). A CMCL command must be executed before any outline can be drawn and filled.

5.3.11 BLOCK TRANSFER [CBLK]

A CBLK command transfers data in a work area specified by the parameter registers ASD and AED. Data can be transferred between the CPU and the FGA, the mode of which is specified by the mode register MOD1 (DMA transfer or programmed transfer controlled by the -CS signal).

When a CBLK command is executed, data are transferred repeatedly and the handshake between CPU and FGA is controlled by the -CS or -DACK signal and -DTACK signal.

When the last piece of data is transferred, the FGA sets the interrupt bit INTMB, and generates an interrupt request to inform the CPU of the end of the transfer.

5.4 SETTING PARAMETERS

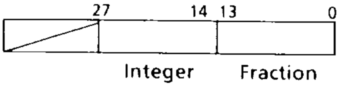
5.4.1 SETTING PARAMETERS for DRAWING a BEZIER CURVE

Command	:	CBZ1
Parameter registers	:	BX0, BX1, BX2, BX3 BY0, BY1, BY2, BY3

To draw a Bezier curve, set the parameter registers BX0~3 and BY0~3 to the following values:

BX0~3	:	X coordinates of the reference points P0 ~ P3 + 0.5
BY0~3	:	Y coordinates of the reference points P0 ~ P3 + 0.5

Each of these parameter registers is composed of a 14 bit integer part and a 14 bit fraction part.



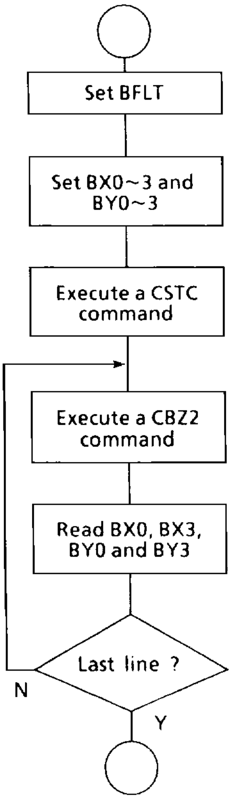
Command	:	CBZ2
Parameter registers	:	BX0, BX1, BX2, BX3 BY0, BY1, BY2, BY3 BFLT

The parameter registers BX0~3 and BY0~3 should be set to the same values as for the CBZ1 command.

CBZ2 commands are used to divide a Bezier curve into several lines. A Bezier curve is approximated by these lines as determined by the flatness parameter. The flatness of a Bezier curve is specified by the parameter register BFLT as follows next page:

FLATNESS	VALUE OF BFLT
1/2	11111
1	11110
2	11100
4	11000
8	10000
16	00000

A Bezier curve can be divided by a CBZ2 command in the following algorithm:



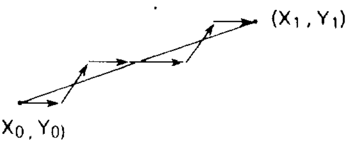
BX0 and BY0 (BX3 and BY3) indicate values equal to 0.5 plus the X or Y coordinate of the beginning (end) of each approximating line.

A line is defined to be the last line if:
- The STSE bit of the status register is set (stack is empty), or
- Both the INTB2 bit and the INTB1 bit of the interrupt register IR are set after a CBZ2 command is executed.

5.4.2 SETTING PARAMETERS for DRAWING a LINE

Command : CDDA1
Parameter registers : DDAR0, DDAR3, DDAR4, DDAC
Mode register bits : DDAMD0, DDAMD1, DDAMD2, DDAMD4

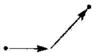
Beginning (X_0, Y_0)
End (X_1, Y_1)



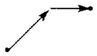
$\Delta X = X1 - X0$	$DDAMD1 = 0$	$\Delta X \geq 0$
	$DDAMD1 = 1$	$\Delta X < 0$
$\Delta Y = Y1 - Y0$	$DDAMD2 = 0$	$\Delta Y \geq 0$
	$DDAMD2 = 1$	$\Delta Y < 0$
$\Delta L = \mathrm{Max}(\lvert \Delta X \rvert, \lvert \Delta Y \rvert)$		$DDAMD0 = 0 \quad \lvert \Delta X \rvert \geq \lvert \Delta Y \rvert$
		$DDAMD0 = 1 \quad \lvert \Delta X \rvert < \lvert \Delta Y \rvert$
$\Delta S = \mathrm{Min}(\lvert \Delta X \rvert, \lvert \Delta Y \rvert)$		$DDAR0 = 2\Delta S - \Delta L$
		$DDAR3 = 2\Delta S$
		$DDAR4 = 2\Delta S - 2\Delta L$
		$DDAC = \Delta L$

The DDAMD4 bit determines whether a coordinate with a fraction of 0.5 should be rounded upward or downward according to the axial direction of the shorter displacement; i.e., if DDMA4 = 0, a coordinate with a fraction of 0.5 will be rounded downward (upward) to the opposite direction to the shorter displacement, and if DDAMD4 = 1, it will be rounded upward (downward) to the same direction as the shorter displacement.

Example)



DDAMD4 = 0



DDAMD4 = 1

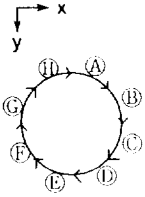
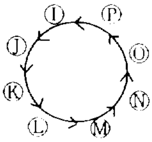
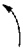
In the above description, the line begins and ends at integral coordinates. If the coordinates contain a fraction, dx, dy steps for the beginning and end points must be provided from the host processor.

5.4.3 SETTING PARAMETERS for DRAWING a CIRCLE

Command	:	CDDA2
Parameter registers	:	DDAR0, DDAR1, DDAR2, DDAR3, DDAR4, DDAC
Mode register bits	:	DDAMD0, DDAMD1, DDAMD2, DDAMD3, DDAMD4, DDAMD7

Each CDDA2 command generates a 1/8 circle.

A 1/8 circle can be drawn in 16 forms as shown below, and defined by the mode register bits DDAMD0, DDAMD1, DDAMD2, and DDAMD3. A 1/8 circle may start or end at a given point on the x or y axis (based on its center as the origin). The former is categorized as a case I circle, and the latter a case II circle.

	NO.		DDAMD0	DDAMD1	DDAMD2	DDAMD3	CASE
	Ⓐ		0	0	0	0	I
	Ⓑ		1	0	0	0	II
	Ⓒ		1	1	0	0	I
	Ⓓ		0	1	0	0	II
	Ⓔ		0	1	1	0	I
	Ⓕ		1	1	1	0	II
	Ⓖ		1	0	1	0	I
	Ⓗ		0	0	1	0	II
	Ⓘ		0	1	0	1	I
	Ⓙ		1	1	0	1	II
	Ⓚ		1	0	0	1	I
	Ⓛ		0	0	0	1	II
	Ⓜ		0	0	1	1	I
	Ⓝ		1	0	1	1	II
	Ⓞ		1	1	1	1	I
	Ⓟ		0	1	1	1	II

Set DDAMD4 and DDAMD7 to 0.

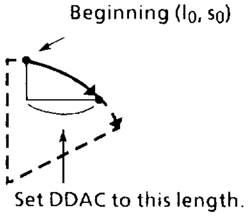
Set DDAC to the coordinates dx, dy.

Set DDAR3 to 4.

Set DDAR4 to 8.

Set DDAR0 – DDAR2 as follows according to the type of a 1/8 circle to be generated (case I or II):

Case I



l_0 and s_0 indicate the x – y displacements from the beginning to the end of this 1/8 circle, based on its center as the origin. R : Radius

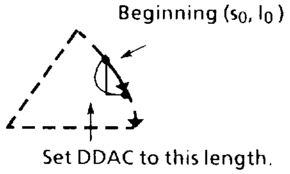
In this case, the x displacement is larger than the y displacement.

$$DDAR0 = 2(l_0^2 + S_0^2 - R^2) + 4|l_0| - 2|S_0| + 3$$

$$DDAR1 = 4|l_0| + 2$$

$$DDAR2 = 4(|l_0| - |S_0|) + 2$$

Case II

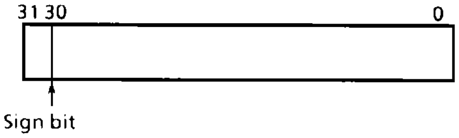


Each of the registers DDAR0~DDAR7 consists of 32 bits, and uses the representation of sign plus 2's complement.

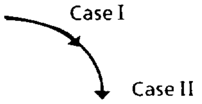
$$DDAR0 = 2(l_0^2 + S_0^2 - R^2) - 4|l_0| + 2|S_0| + 3$$

$$DDAR1 = -4|l_0| + 2$$

$$DDAR2 = 4(|S_0| - |l_0|) + 2$$



5.4.4 ARCS CONSISTING of MORE THAN ONE 1/8 CIRCLE



An arc can be drawn using several 1/8 circles. It should be composed of case I 1/8 circles and case II 1/8 circles that alternate with each other.

If a case I 1/8 circle is already drawn, the case II 1/8 circle that follows it can be drawn using the values remaining in the parameter registers DDAR0–DDAR2. Use the following formulas to calculate the initial parameters of the case II 1/8 circle:

DDAR0 = DDAR0 old + 1/2DDAR2 old − DDAR1 old + 1
DDAR1 = DDAR2 old − DDAR1 old + 2
DDAR2 = DDAR2 old

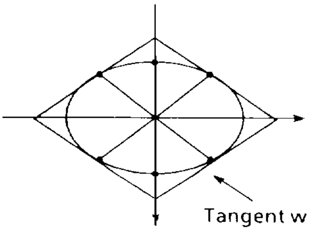
Similarly, if a case II 1/8 circle is already drawn, the case I 1/8 circle that follows it can be drawn using the values remaining in the parameter registers DDAR0–DDAR2.

The following formulas should be used to calculate the initial parameters of the case I 1/8 circle:

DDAR0 = DDAR0 old + DDAR1 old − DDAR2 old
DDAR1 = DDAR1 old
DDAR2 = 2DDAR1 old − DDAR2 old

5.4.5 SETTING PARAMETERS for DRAWING an ELLIPSE (Ellipses parallel to the x and y axes)

Each CDDA2 command generates a 1/8 ellipse. 1/8 ellipse is any of the following 8 sections of an ellipse :



An ellipse must satisfy the following relation : $ax^2 + by^2 + c = 0$

(x and y are the x and y coordinates whose origin is the center of the ellipse).

Point of contact with a ± 1 gradient

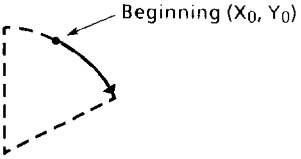
$$x = \pm \sqrt{\frac{-cb}{a\,(a+b)}}$$

$$y = \frac{a}{b}\,x$$

Set the mode register bits DDAMD0~4 and 7 and the parameter register DDAC in the same manner as in setting the parameters of a 1/8 circle.

DDAR0~4 should be set as follows according to the type of a 1/8 ellipse to be generated :

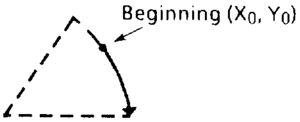
A case I 1/8 elliptic arc that starts from the y axis :



(If the ellipse starts from the x axis, the same formulas will apply, except that "a" should be read as "b" and "X_0" as "Y_0" and vice versa. a ↔ b, $X_0 \leftrightarrow Y_0$)

$DDAR0 = 2\,(ax_0^2 + by_0^2 + c) + 4a\,|x_0| - 2b\,|y_0| + 2a + b$
$DDAR1 = 2a\,(2\,|x_0| + 1)$
$DDAR2 = 2a\,(2\,|x_0| + 1) - 4b\,|y_0|$
$DDAR3 = 4a$
$DDAR4 = 4a + 4b$

A case II 1/8 elliptic arc that ends on the y axis :



(If the ellipse ends on the x axis, the same formulas will apply, except that "a" should be read as "b" and "X_0" as "Y_0", and vice versa. a ↔ b, $X_0 \leftrightarrow Y_0$)

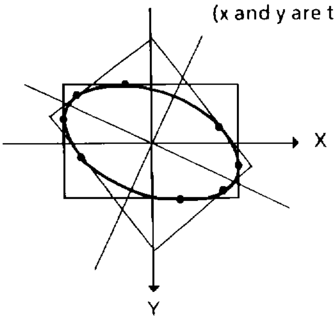
$DDAR0 = 2\,(ax_0^2 + by_0^2 + c) + 2a\,|x_0| - 4b\,|y_0| + 2a + b$
$DDAR1 = 2b\,(-2\,|y_0| + 1)$
$DDAR2 = 4a\,|x_0| + 2b\,(-2\,|y_0| + 1)$
$DDAR3 = 4b$
$DDAR4 = 4a + 4b$

5.4.6 ELLIPSES NOT PARALLEL to X or Y AXIS (Ellipses not parallel to x or y axis)

Each CDDA2 command generates a 1/8 ellipse.

As shown below, an ellipse is divided into 8 sections as determined by the tangent lines parallel to the x and y axes and the ± 1 tangent lines.

Ellipses not parallel to x or y axis : $ax^2 + 2bxy + cy^2 + d = 0$
(x and y are the x and y coordinates whose origin is the center of the ellipse)



Point of contact with horizontal

$$x = \pm b\sqrt{\frac{-d}{a(ac-b^2)}}$$

$$y = \mp a\sqrt{\frac{-d}{a(ac-b^2)}}$$

Point of contact with vertical

$$x = \pm c\sqrt{\frac{-d}{c(ac-b^2)}}$$

$$y = \mp b\sqrt{\frac{-d}{c(ac-b^2)}}$$

Point of contact with a -1 gradient

$$x = \pm (c-b)\sqrt{\frac{-d}{a(c-b)^2 + 2b(a-b)(c-b) + c(a-b)^2}}$$

$$y = \pm (a-b)\sqrt{\frac{-d}{a(c-b)^2 + 2b(a-b)(c-b) + c(a-b)^2}}$$

Point of contact with a + 1 gradient

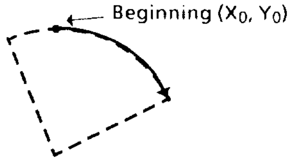
$$x = \pm (c+b)\sqrt{\frac{-d}{a(b+c)^2 - 2b(a+b)(c+b) + c(a+b)^2}}$$

$$y = \pm (a+b)\sqrt{\frac{-d}{a(b+c)^2 - 2b(a+b)(c+b) + c(a+b)^2}}$$

Set the mode register bits DDAMD0~4 and the parameter register DDAC in the same manner as in setting the parameters of a 1/8 parallel ellipse. Set DDAMD7 to 1. DDAR0~5 should be set as follows according to the type of a 1/8 to be generated:

A case I 1/8 ellipses (1/8 ellipses are classified into 16 types. The classification is the same as that of 1/8 circles.)

Type Ⓐ in Case Ⅰ



(No. of Ⓐ~Ⓟ defined as same as a 1/8 ellipse.)

$$\begin{aligned}
DDAR0 &= 2(ax_0^2 + bx_0 y_0 + cy_0^2 + d) \\
&\quad + 2a(2x_0 + 1) + b(x_0 + 2y_0 + 1) + c(2y_0 + 1) \\
DDAR1 &= 2a(2x_0 + 1) + b(2y_0 + 1) \\
DDAR2 &= 2a(2x_0 + 1) + b(2x_0 + 2y_0 + 1) - 4cy_0 \\
DDAR3 &= 4a \\
DDAR4 &= 4a + 4b + 4c \\
DDAR5 &= 4a + 2b
\end{aligned}$$

Type Ⓔ Use the same formulas except the following changes:

$x_0 \rightarrow -x_0$

$y_0 \rightarrow -y_0$

Type Ⓘ Use the same formulas except the following changes:

$x_0 \rightarrow -x_0$

$b \rightarrow -b$

Type Ⓜ Use the same formulas except the following changes:

$y_0 \rightarrow -y_0$

$b \rightarrow -b$

Type Ⓒ Use the same formulas except the following changes:

$x_0 \rightarrow y_0$

$y_0 \rightarrow -x_0$

$a \leftrightarrow c$

$b \rightarrow -b$

Type Ⓖ Use the same formulas except the following changes:

$x_0 \rightarrow -y_0$

$y_0 \rightarrow x_0$

$a \leftrightarrow c$

$b \rightarrow -b$

Type Ⓚ Use the same formulas except the following changes:

$x_0 \rightarrow y_0$

$y_0 \rightarrow x_0$

$a \leftrightarrow c$

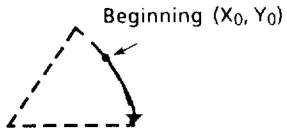
Type Ⓞ Use the same formulas except the following changes:

$x_0 \rightarrow -y_0$

$y_0 \rightarrow -x_0$

$a \leftrightarrow c$

Type Ⓑ in Case II



$$\begin{aligned}
DDAR0 &= 2(ax_0^2 + bx_0 y_0 + cy_0^2 + d) \\
&\quad + a(2x_0 + 1) + b(2x_0 + y_0 + 1) + 2c(2y_0 + 1) \\
DDAR1 &= 2c(2y_0 + 1) + b(2x_0 + 1) \\
DDAR2 &= 2c(2y_0 + 1) + b(2x_0 + 2y_0 + 1) - 4ax_0 \\
DDAR3 &= 4c \\
DDAR4 &= 4a + 4b + 4c \\
DDAR5 &= 2b + 4c
\end{aligned}$$

Type Ⓕ Use the same formulas except the following changes:

$x_0 \rightarrow -x_0$

$y_0 \rightarrow -y_0$

Type Ⓙ Use the same formulas except the following changes:

$x_0 \rightarrow -x_0$

$b \rightarrow -b$

Type Ⓝ Use the same formulas except the following changes:

$y_0 \rightarrow -y_0$

$b \rightarrow -b$

Type Ⓓ Use the same formulas except the following changes:

$x_0 \rightarrow y_0$

$y_0 \rightarrow -x_0$

$a \leftrightarrow c$

$b \rightarrow -b$

Type Ⓗ Use the same formulas except the following changes:

$x_0 \rightarrow -y_0$

$y_0 \rightarrow x_0$

$a \leftrightarrow c$

$b \rightarrow -b$

Type Ⓛ Use the same formulas except the following changes:

$x_0 \rightarrow y_0$

$y_0 \rightarrow x_0$

$a \leftrightarrow c$

Type Ⓠ Use the same formulas except the following changes:

$x_0 \rightarrow -y_0$

$y_0 \rightarrow -x_0$

$a \leftrightarrow c$

5.5 MEMORY ORGANIZATION

5.5.1 DATA ORGANIZATION of MEMORY

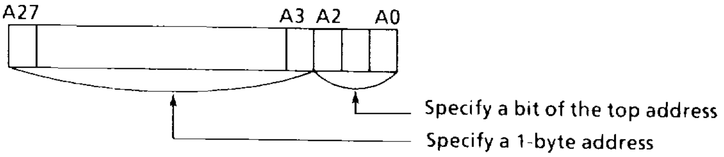
The FGA supports 2 buffer memories A and B, each of which has an 8-bit bus. As a result, the buffer memories are accessed by the FGA on a byte basis.

Data are transferred between the CPU and the FGA using a 32-bit or 16-bit bus. The buffer memories are accessed by the CPU either by dot (for dot-matrix drawing) or by byte (for random access, block transfer and transfer of the results of fill commands).

Data are transferred between the CPU and the buffer memories in words (over a 16-bit bus) or long words (32-bit bus) from a 1-byte memory address specified.

A pair of current pointers are used by the CPU to specify buffer memory addresses. The current pointer L indicates the x coordinates of memory addresses and the current pointer H indicates their y coordinates. The 3 bits A0 to A2 of the current pointer L are used to specify a particular bit of the top address.

Address register



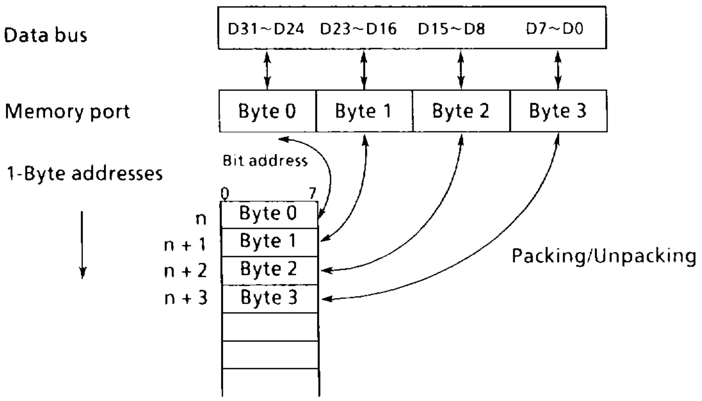
All bits from A0 through A27 are used for outlining and filling. In the data transfer, only the bits A3 to A27 (25 bits) are used to specify a 1-byte address.

The FGA converts words (or long words) to and from an equivalent number of bytes (this is called packing and unpacking).

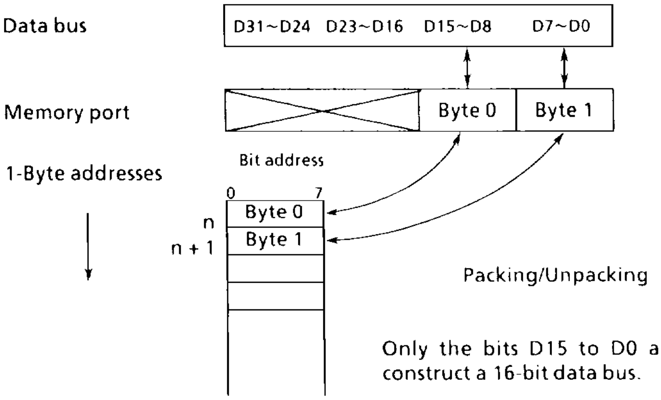
Bit assignment of 1-byte addresses

Each memory address is composed of 8 bits from bit 0 to bit 7. The bit 0 corresponds to the MSB of the data bus between the FGA and the CPU/buffer memories.

32-bit bus (W/-LW signal = low)



16-bit bus (W/-LW signal = high)

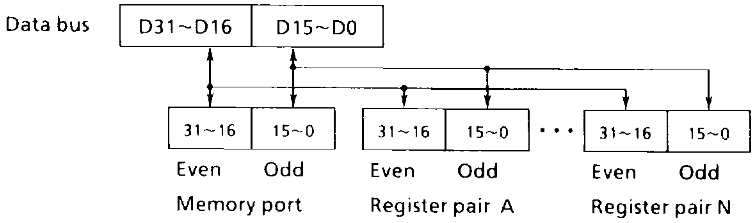


Only the bits D15 to D0 are used to construct a 16-bit data bus.

5.5.2 REGISTER ORGANIZATION

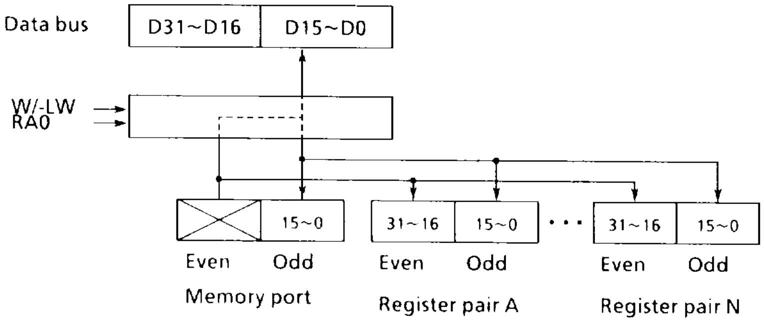
The FGA registers are accessed by the CPU using the register address terminals (RA5-RA0). A register can be accessed only if its number (address) is specified (except in the DMA transfer where a separate memory port is automatically selected). Each of these registers is assigned to a 1-word address, and can be accessed as determined by the width of the bus between the FGA and the CPU.

32-bit bus (W/-LW = low)



If the bus is composed of 32 bits, the CPU will access two 1-word addresses at the same time. As a result, two registers can be accessed if only one address is specified. If an even address n is specified, a pair of registers located in the addresses n and n+1 will be accessed. If an odd address m is specified, registers in the addresses m and m−1 will be accessed.

16-bit bus (W/-LW = high)



In the 16-bit configuration, the bus is composed of the bits D15-D0. This means that only one address can be accessed at one time. Either an even address or odd address must be specified to access each register.

Note however that the memory port is always assigned to an odd address in the 16-bit bus construction.

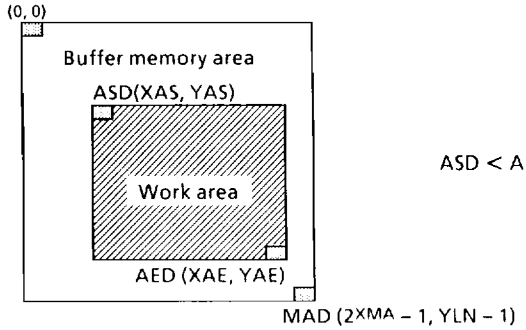
5.6 BUFFER MEMORY and WORK AREA

5.6.1 DEFINING the WORK AREA

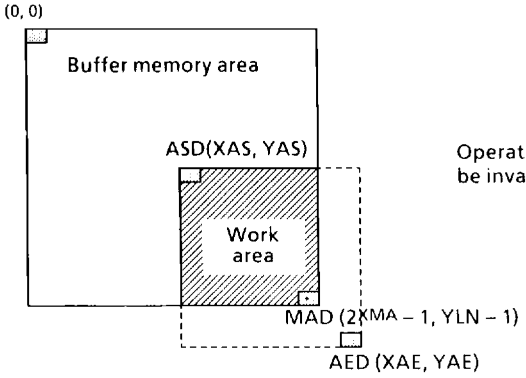
The work area is the active field for all operations including outlining and filling. Also, this area defines data to be transferred or cleared from the buffer memories.

The work area is defined by the parameter registers ASD (Area Start Definition) and AED (Area End Definition).

The work area should be created within the buffer memory area defined by MAD. A work area that extends over the boundary is invalid. All operations outside the MAD may be also invalidated.



ASD < AED



Operations outsides the MAD may be invalidated.

YLN: Y area width
See buffer memory area in the item 4-5-5

5.6.2 WORK AREA for the DATA TRANSFER

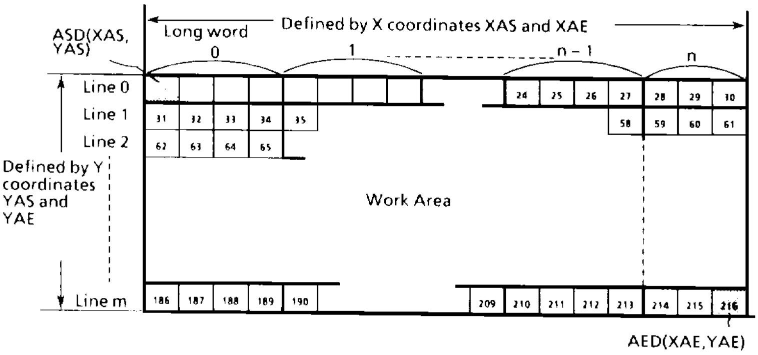
The horizontal range of the work area is defined on a byte basis by the ASD and AED registers. On the other hand, data are transferred to the CPU by word (over the 16-bit data bus) or long word (over the 32-bit data bus). This means that the size of the work area may not be divided by that of data accessible by the CPU. The discrepancy will be 1 byte in the word-based transfer, and 1 to 3 bytes in the long-word-based transfer.

To avoid this problem, the horizontal range of the work area should be defined so it consists of a number of bytes equal to a 4's multiple (or 2's multiple in the 16-bit transfer).

Objects are outlined and filled normally on a work area composed of odd bytes.

Caution should be exercised in attempting a programmed transfer using the chip select (CS) signal. If the work area is sized as shown below and the data bus is composed of 32 bits, the nth transfer cycle results in the transfer of the last 3 bytes on the line 0 **28, 29** and **30** the 1st byte on the line 1 **31**. In this manner, 4 bytes of data will be transferred normally up to the bytes **211**, **212**, **213** and **214** on the last line m. However, the last cycle will cause the FGA to transfer the bytes **215** and **216**, and 2 bytes' undefined data to the CPU.

If a block write transfer is executed under the same conditions, the FGA will write only the existing data of up to **216** .

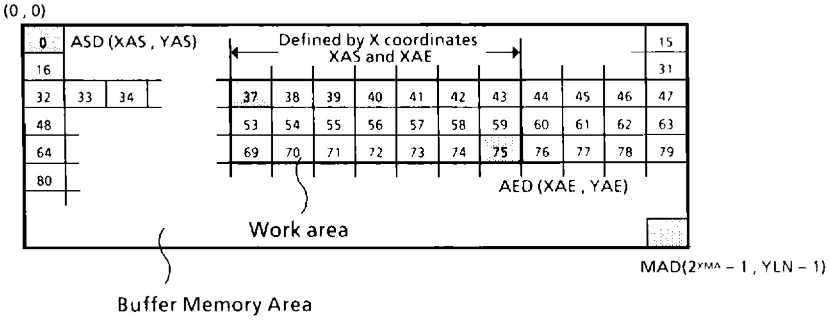


Given conditions:
Bus width: 32 bits
Width of the work area: 4n + 3 bytes

5.6.3 WORK AREA for the DMA TRANSFER

In the DMA transfer, the number of words to be transferred is specified by the DW register to define the active width. Normally, the active width should match the width of the work area specified by the x coordinates XAS and XAE. If the active width is smaller than the width of the work area, a specified number of words will be transferred normally. If the active area is wider than the work area, the shortage will be supplemented by undefined data as described below.

Examples of DMA transfer)



Example 1

- Bus width: 16 bits (W/-LW = 1)
- Number of words to be transferred: 3
- Transfer mode: Line mode (Bit 14 of MOD1 = 1)

  Result of a word-based transfer

  37 38, 39 40, 41 42

Example 2

- Bus width: 16 bits (W/-LW = 1)
- Number of words to be transferred: 4
- Transfer mode: Line mode (Bit 14 of MOD1 = 1)

  Result of a word-based transfer

  37 38, 39 40, 41 42, 43 44

Example 3

- Bus width: 16 bits (W/-LW = 1)
- Number of words to be transferred: 7
- Transfer mode: Line mode (Bit 14 of MOD1 = 1)

  Result of a word-based transfer

  37 38, 39 40, 41 42, 43 44, 45 46, 47 32, 33 34

Example 4

- Bus width: 32 bits (W/-LW = 0)
- Number of words to be transferred: 2
- Transfer mode: Line mode (Bit 14 of MOD1 = 1)

  Result of a word-based transfer

  37 38, 39 40, 41 42 43 44

Example 5

- Bus width: 32 bits (W/-LW = 0)
- Number of words to be transferred: 3
- Transfer mode: Line mode (Bit 14 of MOD1 = 1)

  Result of a long word-based transfer

  37 38 39 40, 41 42 43 44, 45 46 47 32

Example 6

- Bus width: 16 bits (W/-LW = 1)
- Number of words to be transferred: 4
- Transfer mode: Block mode (Bit 14 of MOD1 = 0)

  Result of a word-based transfer

  37 38, 39 40, 41 42, 43 44
  53 54, 55 56, 57 58, 59 60
  69 70, 71 72, 73 74, 75 76

Example 7

- Bus width: 32 bits (W/-LW = 1)
- Number of words to be transferred: 3
- Transfer mode: Block mode (Bit 14 of MOD1 = 0)

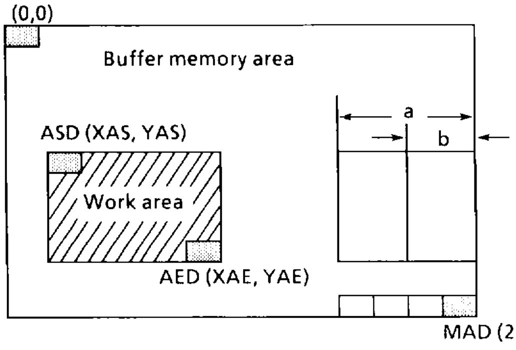
  Result of a word-based transfer

  37 38 39 40 ,41 42 43 44 ,45 46 47 32
  53 54 55 56 ,57 58 59 60 ,61 62 63 48
  69 70 71 72 ,73 74 75 76 ,77 78 79 64

The **Undefined** bytes are not written/read to/from the buffer memories for reading/writing. However, the FGA is accessed by the DMA controller for the number of words specified by the DW register.

5.6.4 WORK AREA for a DIVIDED FONT

To fill a divided font using CFIL2-4 commands, the FGA requires an additional work area to be provided at the right end in the buffer memory area (MAD). This additional work area has the same height (defined by the y coordinates YAS and YAE) as the main work area, and a width of either a word (16 bits) or a long word (32 bits) as determined by the bus width. To process a divided font, the main work area must be located so it will not disturb the additional work area.



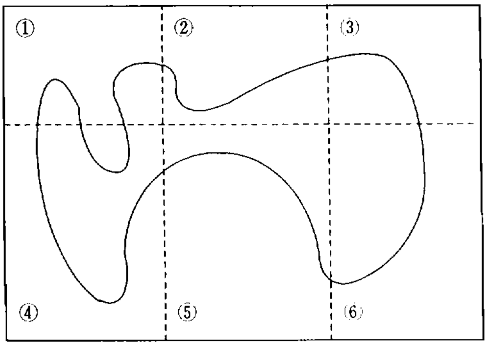
a : Width of an additional work area in the 32-bit configuration

b : Width of an additional work area in the 16-bit configuration

YLN : Y area width
See buffer memory area in the item 5.8.5 (5).

The additional work area is created in the buffer memory A.

One additional work area is repeatedly used for all traverse sections of a divided font. Therefore, the top and the bottom of the main work areas must be unchanged for all of these sections. If the top or the bottom changes, the divided font may not be filled properly.

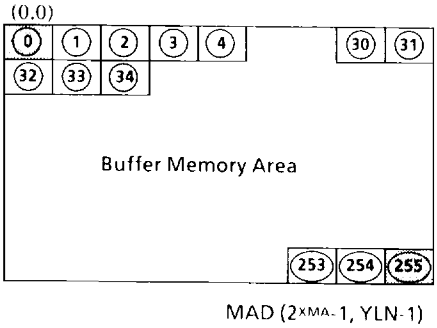


A font divided as shown on the left is filled in the order of ①→②→③→④→⑤→⑥. The sections ①, ② and ③ are filled using the same additional work area, and so their main work areas must be identical in height. Similarly, the top and the bottom of the main work area must be unchanged for the sections ④, ⑤ and ⑥. The height of the main work area can be changed only after all traverse sections (①, ② and ③ or ④, ⑤ and ⑥ in this case) are completed.

The left or the right edge of the main work area can be changed freely.

5.6.5 RANDOM ACCESS and the BUFFER MEMORY AREA

Data in the buffer memory area (MAD) can be randomly accessed by the CPU as specified by the current pointer (CP). Data are accessed either by word (2 bytes) or by long word (4 bytes) according to the bus width.



Over the 32-bit data bus, this buffer memory area can be randomly accessed as follows:

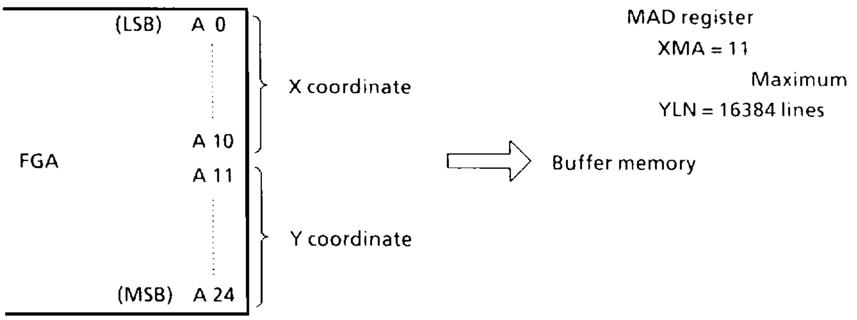
- When CP=0, the bytes **0**, **1**, **2** and **3** will be accessed. In the auto-increment mode, the access will increment the CP value to **4**.
- When CP =**30**, the bytes **30**, **31**, **32** and **33** will be accessed. In the auto-increment mode, the access will increment the CP value to **34**.
- When CP =**253**, the bytes **253**, **254**, **255** and **0** will be accessed. In the auto-increment mode, the access will increment the CP value to **0**.

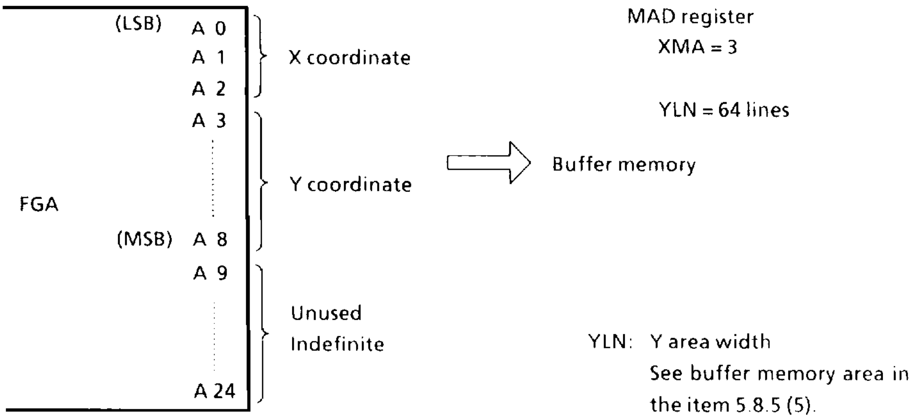
YLN : Y area width
See buffer memory area in
the item 5.8.5 (5).

5.6.6 STRUCTURE of the BUFFER MEMORY ADDRESSES

As mentioned earlier, buffer memory addresses are specified by their x-y coordinates, which are defined by the 25 bits of the current pointers. Up to 11 bits (A0~A10 in the figure below) can be used to indicate an X coordinate, and 14 bits (A11~A24) to indicate a Y coordinate.

The buffer memory area can be sized freely (to a 2's multiple in both the x and y direction) within the capacity of the memory used. The buffer memory area is specified by the parameter register MAD. If an address can be indicated without using the all bits, it will be barrel-shifted (compressed to the LSB), and converted to a linear address. The values in unused bits are undefined.





5.7 RESTRICTIONS

5.7.1 EXECUTION of MORE THAN 1 COMMAND

The FGA commands are located in the command register. A command can be accessed by setting a specific bit allocated to this command. Executing more than one command is disabled by the FGA to protect the internal circuits (except for CSTC and CBZ2 commands). If more than 1 bit are set, the less significant bit (bit of the smaller number) will be selected.

Command register

Bit assignment															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨													
———			CBLK	CMCL	CFIL4	CFIL3	CFIL2	CFIL1	CPTE	CDDA2	CDDA1	CBZ2	CBZ1	CSTC	CRST

→ Priority increases   Highest priority

- The CRST bit has the highest priority. Any other bits will be ignored if set.
- A CSTC command can be coincided with a command from CDDA1 to CBLK (bits 4 – 12).
- If the CSTC bit and the CBZ1 or CBZ2 bit are set at the same time, only the CSTC bit becomes effective.
- A CBLK (bits 4 – 12).
- In any other case where more than 1 bit are set at the same time, the FGA will select the smaller bit, and set the INTIL bit of the interrupt register.

5.7.2 ACCESS by the CPU DURING COMMAND EXECUTION

When a command is being executed, the following access may be attempted by the CPU:

(1) Access to another command
(2) Access to other internal register
(3) Access to the dx, dy port register
(4) Access to the memory port register

(1) Access to another command

If the access to another command is attempted,

- A reset command will be always executed if accessed.
- A CBZ2 or CSTC command will be executed if certain conditions are met.
- Other command will be always executed if a CBZ2 command is currently executed.

In all other cases, the FGA disables the access and sets the INTIL bit of the interrupt register.

(2) Access to other internal register

- Mode registers (MOD1 and MOD2)
  Writing to the mode registers will be disabled. If the writing is attempted, the data may be invalidated.

- Parameter registers

PARAMETER REGISTERS

COMMAND BEING EXECUTED		OPERATION MODE	PARAMETER REGISTERS THAT CANNOT BE WRITTEN	READ OPERATION
CBZ1 or CBZ2	CBZ1	Draw outlines in the buffer memory A/B (Bit 1 of MOD1 = 0)	BSP, BX0, BX1, BX2, BX3, BY0, BY1, BY2, BY3, ASD, AED, CP, DS, MAD	Possible
		Draw noting in the buffer memory A/B (Bit 1 of MOD1 = 1)	BX0, BX1, BX2, BX3, BY0, BY1, BY2, BY3 BSP	Possible
	CBZ2	————————	BX0, BX1, BX2, BX3, BY0, BY1, BY2, BY3 BSP, BFLT	Possible
CDDA1 or CDDA2	CDDA1	Draw outlines in the buffer memory A/B (Bit 1 of MOD1 = 0)	DDAR0, DDAR3, DDAR4, DDAC, ASD, AED, CP, DS, MAD	Possible
		Draw nothing in the buffer memory A/B (Bit 1 of MOD1 = 1)	DDAR0, DDAR3, DDAR4, DDAC	Possible
	CDDA1	Draw outlines in the buffer memory A/B (Bit 1 of MOD1 = 0)	DDAR0, DDAR1, DDAR2, DDAR3, DDAR4, DDAR5, DDAC, ASD, AED, CP, DS, MAD	Possible
		Draw nothing in the buffer memory A/B (Bit 1 of MOD1 = 1)	DDAR0, DDAR1, DDAR2,DDAR3, DDAR4, DDAR5, DDAC	Possible
CFIL1 - 4 or CBLK		Request DMA transfer (Bit 13 of MOD1 = 1)	ASD, AED, CP, DS, MAD, DW	Possible
		Request no DMA transfer (Bit 13 of MOD1 = 0)	ASD, AED, CP, DS, MAD	Possible
CMCL, CPTE or buffer memory access Bit 8 (STMBY) of SR1 = 1		————————	ASD, AED, CP, DS, MAD	Possible

(3) Access to the dx, dy port register

If the access to the dx, dy port register is attempted.

- Writing will be enabled only if the command being executed is CBZ2.
- Reading will be enabled only if the command being executed is CBZ1, DDA1 or DDA2 and the mode register is set to enable the output to the CPU.

In all other cases, the FGA disables the access and sets the INTIL bit of the interrupt register.

(4) Access to the memory port register

- Reading/writing will be enabled if the command being executed is CBZ2.

In all other cases, the FGA disables the access and sets the INTIL bit of the interrupt register.

After setting the INTIL bit, the FGA generates the -DTACK signal to the CPU. This will terminate the bus cycle, and prevent the system from getting hung.

5.7.3 ILLEGAL ACCESS

Access by the CPU

Command being executed / Access by CPU		CBZ1	CBZ2	CDDA 1/2	CPTE	CFIL1 ~ 4	CMCL	CBLK	When no command is being executed
Command register	CRST	○	○	○	○	○	○	○	○
	CSTC	×	×	○	○	○	○	○	○
	CBZ1	×	×	×	×	×	×	×	○
	CBZ2	×	×	○	○	○	○	○	○
	CDDA1/2	×	○	×	×	×	×	×	○
	CPTE	×	○	×	×	×	×	×	○
	CFIL1 ~ 4	×	○	×	×	×	×	×	○
	CMCL	×	○	×	×	×	×	×	○
	CBLK	×	○	×	×	×	×	×	○
Internal registers	R	○	○	○	○	○	○	○	○
	W	○	○	○	○	○	○	○	○
dx, dy port register	R	[ ]	×	[ ]	×	×	×	×	×
	W	×	○	×	×	×	×	×	○
Memory port register	R	×	○	×	×	[ ]	×	[ ]	○
	W	×	○	×	×	×	×	[ ]	○

○ Enabled

× Disabled (Illegal access)

[ ] Enabled or disabled according to the mode selected

- If the access is disabled, the FGA will generate the -DTACK signal to the CPU and set the INTIL bit of the interrupt register. All commands (except RESET) will be ignored if accessed. Reading of an internal register, the memory port or dx, dy port register will result in the reading of undefined data. Writing will be disabled.
- If no command is being executed, the memory port register can be accessed only in the random mode. DMA transfer is disabled.

5.8 REGISTER

5.8.1 DESCRIPTION of the REGISTERS

□ Command register (W)

The FGA supports 13 basic commands including CBZ, DDA and CFIL. To execute a command, set the bit assigned to this command.

□ Mode registers (R/W)

The mode registers specify the mode of operation to be executed by a command. They must be set before a command is executed.

□ Parameter registers (R/W)

Parameter registers are located in the Bezier block, DDA block and Fill block. They indicate parameters necessary to execute a command. They must be set before a command is executed.

10 parameter registers in the Bezier block	–	BSP, BFLT, BX0, BX1, BX2, BX3, BY0, BY1, BY2, BY3
7 parameter registers in the DDA block	–	DDAR0, DDAR1, DDAR2, DDAR3, DDAR4, DDAR5, DDAC
6 parameter registers in the Fill block	–	ASD, AED, CP, DS, MAD, DW

□ Status registers (R)

The status registers serve as status flags that indicate the states of the Bezier, DDA and Fill blocks. Up to 22 states can be indicated to help effective control by the CPU.

□ Interrupt register (R)

It supports 9 types of interrupts to enable various operations such as ending the current command, dividing a line, processing an overflow of the stack, and checking the clipping area. An interrupt is requested when the applicable bit is set. If an interrupt bit is set, the register will generate an interrupt request (-IRQ) signal to allow the CPU to process the interrupt routine. An interrupt can be reset by reading the interrupt register.

□ Interrupt enable register (R/W)

This register enables interrupts on a selective basis. Each bit corresponds to a specific bit of the interrupt register.

□ Memory access port register (R/W)

The buffer memories are accessed by the CPU through the memory port. The buffer memories are accessed when:

(1) The result of a fill command is transferred to the CPU;

(2) A block transfer is executed; or

(3) Random data access is selected.

□ dx, dy port register (R/W)

This register is used by the CPU to read/write displacement data (dx, dy) from/to the buffer memories. To access any of these registers, the CPU must specify its register number using the RA5 – RA0 terminals.

5.8.3 LIST of the REGISTERS

Register NO.	Register name	Abbreviation	R/W	Bit Assignment															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
01	Command register	CMD	W				CBLK	CMCL	CFIL4	CFIL3	CFIL2	CFIL1	CPTE	CDDA2	CDDA1	CBZ2	CBZ1	CSTC	CRST
02	Status register 2	SR2	R							STCL9	STCL8	STCL7	STCL6	STCL5	STCL4	STCL3	STCL2	STCL1	STCL0
03	Status register 1	SR1	R	STSE	STFOV	STNCL					STMBY	STBLK	STMCL	STFL	STPTE	STD2	STD1	STB2	STB1
04	Interrupt register	IR	R								INTIL	INTPE	INTCP	INTAC	INTSO	INTMB	INTD	INTB2	INTB1
05	Interrupt enable register	IER	R/W								INEIL	INEPE	INECP	INEAC	INESO	INEMB	INED	INEB2	INEB1
06	Mode register 2	MOD2	R/W	MDDA7			MDDA4	MDDA3	MDDA2	MDDA1	MDDA0							MMW1	MMW0
07	Mode register 1	MOD1	R/W	MDB	MDL	MDR		MMA1	MMA0	MMC	MMC1	MMC0	MOP1	MOP0	MFL1	MFL0	MXY	MDR1	MDR0
08	Memory port (H)	MP	R/W	MPH															
09	Memory port (L)		R/W	MPL															

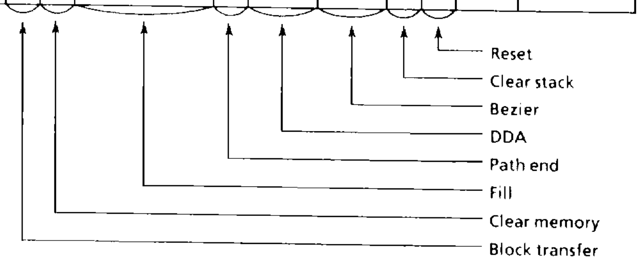
<table>
<tr><th rowspan="2">Register No.</th><th rowspan="2">Register Name</th><th rowspan="2">Abbreviation</th><th rowspan="2">R/W</th><th colspan="16">Bit Assignment</th></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="16"></td></tr>
<tr><td>0B</td><td>dx, dy port</td><td>DXYP</td><td>R/W</td><td colspan="12">▨</td><td>DYS</td><td>DYO</td><td>DXS</td><td>DXO</td></tr>
<tr><td>⋮</td><td></td><td></td><td></td><td colspan="16"></td></tr>
<tr><td>10</td><td>Work area start definition (H)</td><td rowspan="2">ASD</td><td>R/W</td><td colspan="2">▨</td><td colspan="14">YAS</td></tr>
<tr><td>11</td><td>Work area start definition (L)</td><td>R/W</td><td colspan="5">▨</td><td colspan="11">XAS</td></tr>
<tr><td>12</td><td>Work area end definition (H)</td><td rowspan="2">AED</td><td>R/W</td><td colspan="2">▨</td><td colspan="14">YAE</td></tr>
<tr><td>13</td><td>Work area end definition (L)</td><td>R/W</td><td colspan="5">▨</td><td colspan="11">XAE</td></tr>
<tr><td>14</td><td>Current pointer (H)</td><td rowspan="2">CP</td><td>R/W</td><td colspan="2">▨</td><td colspan="14">YCP</td></tr>
<tr><td>15</td><td>Current pointer (L)</td><td>R/W</td><td colspan="2">▨</td><td colspan="11">XCP</td><td colspan="3">BCP</td></tr>
<tr><td>16</td><td>Drawing start (H)</td><td rowspan="2">DS</td><td>R/W</td><td colspan="2">▨</td><td colspan="14">YDS</td></tr>
<tr><td>17</td><td>Drawing start (L)</td><td>R/W</td><td colspan="2">▨</td><td colspan="11">XDS</td><td colspan="3">BDS</td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="16"></td></tr>
<tr><td>19</td><td>Memory area definition</td><td>MAD</td><td>R/W</td><td colspan="12">▨</td><td colspan="4">XMA</td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="16"></td></tr>
<tr><td>1B</td><td>DMA word count</td><td>DW</td><td>R/W</td><td colspan="6">▨</td><td colspan="10">WN</td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="16"></td></tr>
</table>

Register No.	Register Name	Abbreviation	R/W	Bit Assignment															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1E	Bezier stack pointer	BSP	R/W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	SP				
1F	Bezier flatness	BFLT	R/W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	FLT				
20	Bezier X0 (H)	BX0	R/W	▨	▨	▨	▨	X0H											
21	Bezier X0 (L)		R/W	X0L															
22	Bezier X1 (H)	BX1	R/W	▨	▨	▨	▨	X1H											
23	Bezier X1 (L)		R/W	X1L															
24	Bezier X2 (H)	BX2	R/W	▨	▨	▨	▨	X2H											
25	Bezier X2 (L)		R/W	X2L															
26	Bezier X3 (H)	BX3	R/W	▨	▨	▨	▨	X3H											
27	Bezier X3 (L)		R/W	X3L															
28	Bezier Y0 (H)	BY0	R/W	▨	▨	▨	▨	Y0H											
29	Bezier Y0 (L)		R/W	Y0L															
2A	Bezier Y1 (H)	BY1	R/W	▨	▨	▨	▨	Y1H											
2B	Bezier Y1 (L)		R/W	Y1L															
2C	Bezier Y2 (H)	BY2	R/W	▨	▨	▨	▨	Y2H											
2D	Bezier Y2 (L)		R/W	Y2L															

Register No.	Register Name	Abbreviation	R/W	Bit Assignment															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2E	Bezier Y3 (H)	BY3	R/W	▨	▨	▨	▨	Y3H											
2F	Bezier Y3 (L)		R/W	Y3L															
30	DDA 0 (H)	DDAR0	R/W	ACC0H															
31	DDA 0 (L)		R/W	ACC0L															
32	DDA 1 (H)	DDAR1	R/W	ACC1H															
33	DDA 1 (L)		R/W	ACC1L															
34	DDA 2 (H)	DDAR2	R/W	ACC2H															
35	DDA 2 (L)		R/W	ACC2L															
36	DDA 3 (H)	DDAR3	R/W	CST0H															
37	DDA 3 (L)		R/W	CST0L															
38	DDA 4 (H)	DDAR4	R/W	CST1H															
39	DDA 4 (L)		R/W	CST1L															
3A	DDA 5 (H)	DDAR5	R/W	CST2H															
3B	DDA 5 (L)		R/W	CST2L															
⋮																			
3F	DDA counter	DDAC	R/W	CUNT															

5.8.3 COMMAND REGISTER

Bit Assignment																R/W	Register No.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
———			CBLK	CMCL	CFIL4	CFIL3	CFIL2	CFIL1	CPTE	CDDA2	CDDA1	CBZ2	CBZ1	CSTC	CRST	W	01 (hex)



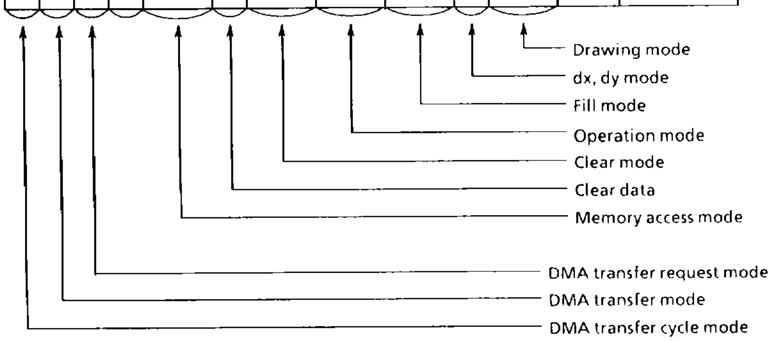
BITS		OPERATION
0	Reset	Cause a software reset
1	Clear stack	Clear the stack pointer
2	Bezier1	Generate dx, dy of a Bezier curve
3	Bezier2	Divide a Bezier curve (without generating dx, dy)
4	DDA1	Generate dx, dy of a straight line
5	DDA2	Generate dx, dy of a circle or ellipse
6	Patch end	Define the end of an outline to be closed
7	Fill 1	Fill a non-divided font
8	Fill 2	Fill a divided font (right sections)
9	Fill 3	Fill a divided font (left sections)
10	Fill 4	Fill a divided font (remaining sections)
11	Clear memory	Clear data from the buffer memories
12	Block transfer	Transfer data between the buffer memories and the CPU

Commands are not decoded by the FGA. They must be controlled by the CPU so they will not be duplicated.

5.8.4 MODE REGISTERS

(1) MODE REGISTER 1 [MOD1]

Bit Assignment																R/W	Register No.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
			▨														
MDB	MDL	MDR		MMA1	MMA0	MMC	MMC1	MMC0	MOP1	MOP0	MFL1	MFL0	MXY	MDR1	MDR0	R/W	07 (hex)



All bits are reset to 0.

□ The following bits indicate the mode of drawing outlines that result from displacements generated by Bezier1, DDA1 and/or DDA2 commands or provided from the CPU :

BITS		OPERATION
1	0	
0	0	Draw outlines in the buffer memory A, and draw outlines including special points in the buffer memory B.
0	1	Draw outlines in the buffer memory A, and draw nothing in the buffer memory B.
1	x	No operation.

If no operation is selected, nothing is drawn in the buffer memories, but the Bezier block and the DDA block will operate normally to generate displacement data. No operation will not affect the dx, dy mode specified by the bit 2.

[ ] The following bit determines the output to the CPU of displacements (dx, dy) generated by Bezier1, DDA1 and/or DDA2 commands:

BITS	OPERATION
2	
0	Not generate dx, dy to the CPU.
1	Generate dx, dy to the CPU.

[ ] The following bits specify the mode of filling and processing the result:

BITS		OPERATION
4	3	
0	0	Determine the area to be filled using data in the buffer memories A and B, and write the result to the buffer memory B. The buffer memory A will be unchanged.
0	1	Determine the area to be filled using data in the buffer memories A and B, and write the result to the buffer memory B and the CPU. The buffer memory A will be unchanged.
1	0	Determine the area to be filled using data in the buffer memories A and B, and write the result to the CPU. Both the buffer memories A and B will be unchanged.
1	1	Determine the area to be filled using data in the buffer memories A and B, and write the result to the CPU. Both the buffer memories A and B will be cleared.

The "10" mode provides the highest speed because the operation involves only the reading of the memories A and B.

[ ] The following bits determines the area to be filled by a fill command:

BITS		OPERATION
6	5	
0	0	Fill an outline and its interior.
0	1	Fill the interior of an outline.
1	0	Fill an outline and its exterior.
1	1	Fill the exterior of an outline.

□ The following bits specify the mode of clearing memories :

BITS		OPERATION
8	7	
0	0	Clear the memories A and B.
0	1	Clear the memory A.
1	0	Clear the memory B.
1	1	No operation

□ The following bit determines how data should be cleared by a CMCL (clear memory) command or a fill command involving a clear process:

BITS	OPERATION
9	
0	Clear data to the "0" state (blank).
1	Clear data to the "1" state (filled).

□ The following bits specify the mode of access to the buffer memory A/B in the block transfer or random access mode (in the latter mode, only the bit 11 is valid) :

BITS		OPERATION
11	10	
0	0	Read the memory B.
0	1	Write the memory B.
1	0	Read the memory A.
1	1	Write the memory A.

□ The following bit specifies whether or not the direct memory access (DMA) should be requested for data output or blocks transfer :

BITS	OPERATION
9	
0	Not request DMA
1	Request DMA

If the DMA is not requested, data transfer to the CPU should be programmed using the chip select (-CS) signal.

□ The following bit determines whether data should be accessed by line or block in a DMA transfer :

BITS	
14	OPERATION
0	Block
1	Line

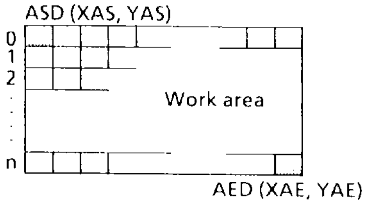
This bit is effective only if a DMA transfer is requested (bit 13 = 1).

A DMA transfer may be used for a fill command that involves a data transfer to the CPU, or a CBLK command.

In the block mode, a DMA transfer allows the access to a two-dimensional space in the work area specified by the parameter registers ASD and AED. In a line mode, data can be accessed only for a specific line in the work area.

The block mode or line mode can be selected for the following commands:

CFIL	:	Block mode
CBLK	:	Block mode/Line mode



Note that a block transfer can be executed either in the block mode or line mode. Given a work area as shown on the left, the whole data can be transferred by a CBLK command line by line only if the top address of each line is specified by the ASD register.

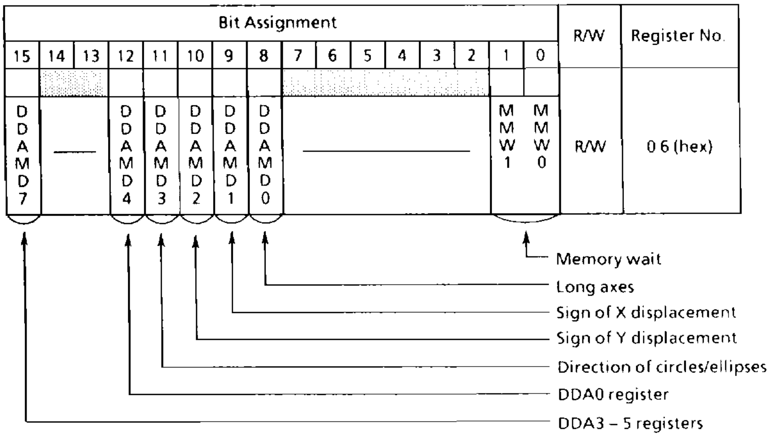
The block mode is automatically selected for a fill command or block transfer command that specifies a programmed transfer using the chip select (-CS) signal.

□ The following bit determines the cycle of a DMA transfer :

BITS	
15	OPERATION
0	Single mode
1	Burst mode

This bit is valid only when the DMA transfer request mode is selected (bit 13 = 1). In the single mode, the -DREQ signal is generated each time a transfer cycle is completed. In the burst mode, the -DREQ signal is kept low until a specified number of words (or long words) are transferred.

(2) MODE REGISTER 2 [MOD2]



Bit Assignment																R/W	Register No.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
DDAMD7	—		DDAMD4	DDAMD3	DDAMD2	DDAMD1	DDAMD0	—						MMW1	MMW0	R/W	06(hex)

All bits are reset to 0.

□ The following bits determines the wait mode for the FGA to access the buffer memories:

BITS		OPERATION
1	0	
0	0	No wait
0	1	Wait for 1 clock cycle
1	0	Wait for 2 clock cycles
1	1	Wait for 3 clock cycles

The other bits indicate DDA operation modes.

□ The following bit specifies the long axes to be determined by a DDA1, 2 command :

BITS	OPERATION
8	
0	$|\Delta X| \geq |\Delta Y|$
1	$|\Delta X| < |\Delta Y|$

□ The following bit specifies the sign of an X displacement to be determined by a DDA1, 2 command :

BITS	OPERATION
9	
0	△X ≥ 0
1	△X < 0

□ The following bit specifies the sign of a Y displacement to be determined by a DDA1, 2 command :

BITS	OPERATION
10	
0	△Y ≥ 0
1	△Y < 0

□ The following bit specifies the direction of a circle or ellipse to be drawn by a DDA2 command :

BITS	OPERATION
11	
0	Right turn
1	Left turn

□ The following bit determines the way of judging value of DDAR0 for DDA steps :

BITS	OPERATION
12	
0	DDAR0 > 0
1	DDAR0 ≥ 0

□ The following bit specifies which constant register(s) should be used for a DDA2 command :

BITS	OPERATION
15	
0	DDAR3 and DDAR4
1	DDAR3, DDAR4 and DDAR5

This bit should be set to "1" before drawing an ellipse which is not parallel to the axes.

5.8.5 PARAMETER REGISTERS

(1) WORK AREA START DEFINITION REGISTER [ASD]

This register indicates the x, y coordinates of the beginning of a work area to be used for plotting, filling, or processing data to be transferred or cleared.

Bit Assignment						R/W	Program Unit	Register No.
15	········	11	10	········	0			
▨	········	▨		········		R/W	Byte	11 (hex)
——			XAS					

XAS: X-Work Area Start

The following condition must be set :

0 ≦ XAS < XAE

All bits are reset to 0

Bit Assignment					R/W	Program Unit	Register No.
15	14	13	········	0			
▨	▨		········		R/W	Line	10 (hex)
——		YAS					

YAS: Y-Work Area Start

The following condition must be set :

0 ≦ YAS ≦ YAE

All bits are reset to 0

(2) WORK AREA END DEFINITION [AED] REGISTER

This register indicates the x, y coordinates of the end of a work area to be used for plotting, filling, or processing data to be transferred or cleared.

Bit Assignment						R/W	Program Unit	Register No.
15	⋯⋯	11	10	⋯⋯	0			
	⋯⋯			⋯⋯				
——			XAE			R/W	Byte	13 (hex)

XAE: X-Work Area End

The following condition must be set:

$XAS + n \leq XAE \leq 2^{XMA} - 1$

where n = 1 if the bus width is 16 bits, and
3 if the bus width is 32 bits.

XAE is reset to 63

Bit Assignment					R/W	Program Unit	Register No.
15	14	13	⋯⋯	0			
			⋯⋯				
——	YAE				R/W	Line	12 (hex)

YAE: Y-Work Area End

The following condition must be set:

$YAS \leq YAE \leq YLN - 1$

YAE is reset to 511

YLN: Y area width
See buffer memory area in the item 5.8.5 (5)

(3) CURRENT POINTER [CP]

This register indicates the x, y coordinates of a memory address to be accessed in the random access mode.

The bits 0 – 2 indicate a particular bit of the address, and are effective only for reading.

This register can also specify the current address of plotting or filling, or the address of data to be transferred or cleared. The CPU can find the current address (x, y coordinates) by reading this register.

Bit Assignment								R/W	Program Unit	Register No.
15	14	13	········	3	2	1	0			
▨	▨		········					R/W	BCP: —— XCP: Byte	15 (hex)
——		XCP			BCP					

XCP: X-Current Pointer
BCP: Bit-Current Pointer
BCP is valid only for reading.

The following condition must be set :

$0 \leq XCP \leq 2^{XMA}-1$

All bits are reset to 0

Bit Assignment					R/W	Program Unit	Register No.
15	14	13	········	0			
▨	▨		········		R/W	Line	14 (hex)
——		YCP					

YCP: Y-Current Pointer

The following condition must be set :

$0 \leq YCP \leq YLN-1$

All bits are reset to 0

YLN: Y area width
See buffer memory area in the item 5.8.5 (5).

In the auto-increment mode, the current pointer is incremented by 4 bytes (32 bits) or 2 bytes (16 bits) as determined by the bus width. Once the current pointer is set, data can be accessed continuously from the set address.

(4) DRAWING START ADDRESS REGISTER [DS]

This register indicates the x, y coordinates of the beginning of an outline, and a particular bit of its address.

Bit Assignment								R/W	Program Unit	Register No.
15	14	13	···············	3	2	1	0			
▨	▨		···············					R/W	BDS: Bit XDS: Byte	1 7 (hex)
——		XDS			BDS					

BDS: Bit-Drawing Start Address
XDS: X-Drawing Start Address

The following condition must be set:

$0 \leq XDS \leq 2^{XMA}-1$

All bits are reset to 0

Bit Assignment					R/W	Program Unit	Register No.
15	14	13	···············	0			
▨	▨		···············		R/W	Line	1 6 (hex)
——		YDS					

YDS: Y-Drawing Address

The following condition must be set:

$0 \leq YDS \leq YLN-1$

All bits are reset to 0

YLN : Y area width
See buffer memory area in the item 5.8.5 (5).

The end of an outline is defined by a CPTE (Path End) command. An outline can be closed only if its end overlaps with its beginning. The FGA compares the DS register and the current pointer to determine if the outline can be closed. The DS register must be kept at the same value until an outline is closed.

(5) BUFFER MEMORY AREA DEFINITION REGISTER [MAD]

This register defines the size of the buffer memory area.

FGA supports a memory area of maximum 11 bits (2048 bytes) × 14 bits (16384 lines). Only the width is specified by this register because the number of lines is automatically determined by the capacity of the buffer memory.

Bit Assignment							R/W	Register No.
15	....	4	3	2	1	0		
▨	....	▨					R/W	19 (hex)
————————			XMA					

XMA: X-Memory Area

The following condition must be set :

- Non-divided font

  $1 \leqq \mathrm{XMA} \leqq 11$ (YMA number from 12 to 15 is counted as 11)

- Divided font

  16-bit bus width

  $2 \leqq \mathrm{XMA} \leqq 11$

  32-bit bus width

  $3 \leqq \mathrm{XMA} \leqq 11$

XMA is reset to 6 (64 bytes)

Number of lines (YLN) ≤ 16384

$$\frac{M}{2^{XMA}} = \text{YLN (Number of lines)}$$

where M is the capacity of the buffer memory (in bytes).

(6) DMA WORD [DW] COUNTER

This counter specifies the number of words to be transferred in the DMA mode.

This counter becomes effective only when a DMA transfer is requested (the bit 13 of the MOD1 register is set to 1).

It specifies the number of words in the X region between the work clear start (ASD) and the work clear end (AED).

The count varies with different widths of bus to the CPU.

Suppose, there are 16 bytes in the X region between ASD and AED. 8 is specified for the bus width of 16 bits, and 4 for 32 bits.

Bit Assignment													R/W	Register No.
15	········	10	9	8	7	6	5	4	3	2	1	0		
	········													
————		WN											R/W	1B (hex)

WN: Word Number

The following condition must be set :

2 ≦ WN ≦ 1023

(7) BEZIER STACK POINTER [BSP]

This pointer indicates the top address of the stack, which stores the division points of a Bezier curve subdivided by recursive division.

This register is cleared by a stack clear (CSTC) command.

Bit Assignment								R/W	Register No.
15	········	5	4	3	2	1	0		
	········								
————		SP						R/W	1E (hex)

SP: Stack Pointer

All bits are reset to 0

(8) BEZIER FLATNESS [BFLT] REGISTER

This register defines a flatness based on which a Bezier curve should be divided by Bezier 2 (CBZ2) commands.

Bit Assignment								R/W	Register No.
15		5	4	3	2	1	0		
								R/W	1F (hex)
			FLT						

FLT: Flatness Number

6 levels of flatness can be specified as follows :

Flatness	Value to be specified
1/2	1 1 1 1 1
1	1 1 1 1 0
2	1 1 1 0 0
4	1 1 0 0 0
8	1 0 0 0 0
16	0 0 0 0 0

(9) REGISTERS USED for BEZIER COMMANDS

4 reference points (BX0, BY0), (BX1, BY1), (BX2, BY2) and (BX3, BY3) are used to approximate each Bezier curve.

Each coordinate is represented by a 14-bit integer and a 14-bit fraction.

A Bezier curve is divided into lines by Bezier 2 commands. Each line is defined by its beginning (BX0, BY0) and end (BX3, BY3) to be stored into the respective registers described below (1).

BEZIER X0 [BX0] REGISTER

Bit Assignment					R/W	Register No.
15	14	13		0		
					R/W	21 (hex)
X01N (L)		X0FN				

X0FN: X0 Fractional Number

Bit Assignment							R/W	Register No
15	14	13	12	11		0		
							R/W	20 (hex)
———				X0IN (H)				

X0IN: X0 Integer Number

BEZIER Y0 [BY0] REGISTER

Bit Assignment					R/W	Register No.
15	14	13		0		
					R/W	29 (hex)
Y0IN (L)		Y0FN				

Y0FN: Y0 Fractional Number

Bit Assignment							R/W	Register No.
15	14	13	12	11		0		
							R/W	28 (hex)
———				Y0IN (H)				

Y0IN: Y0 Integer Number

BEZIER X1 [BX1] REGISTER

<table>
<tr><td colspan="5">Bit Assignment</td><td rowspan="2">R/W</td><td rowspan="2">Register No.</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>..........</td><td>0</td></tr>
<tr><td></td><td></td><td></td><td>..........</td><td></td><td rowspan="2">R/W</td><td rowspan="2">23 (hex)</td></tr>
<tr><td colspan="2">X1IN (L)</td><td colspan="3">X1FN</td></tr>
</table>

X1FN: X1 Fractional Number

<table>
<tr><td colspan="7">Bit Assignment</td><td rowspan="2">R/W</td><td rowspan="2">Register No.</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>..........</td><td>0</td></tr>
<tr><td>▨</td><td>▨</td><td>▨</td><td>▨</td><td></td><td>..........</td><td></td><td rowspan="2">R/W</td><td rowspan="2">22 (hex)</td></tr>
<tr><td colspan="4">——</td><td colspan="3">X1IN (H)</td></tr>
</table>

X1IN: X1 Integer Number

BEZIER Y1 [BY1] REGISTER

<table>
<tr><td colspan="5">Bit Assignment</td><td rowspan="2">R/W</td><td rowspan="2">Register No.</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>..........</td><td>0</td></tr>
<tr><td></td><td></td><td></td><td>..........</td><td></td><td rowspan="2">R/W</td><td rowspan="2">2B (hex)</td></tr>
<tr><td colspan="2">Y1IN (L)</td><td colspan="3">Y1FN</td></tr>
</table>

Y1FN: Y1 Fractional Number

<table>
<tr><td colspan="7">Bit Assignment</td><td rowspan="2">R/W</td><td rowspan="2">Register No.</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>..........</td><td>0</td></tr>
<tr><td>▨</td><td>▨</td><td>▨</td><td>▨</td><td></td><td>..........</td><td></td><td rowspan="2">R/W</td><td rowspan="2">2A (hex)</td></tr>
<tr><td colspan="4">——</td><td colspan="3">Y1IN (H)</td></tr>
</table>

Y1IN: Y1 Integer Number

BEZIER X2 [BX2] REGISTER

Bit Assignment					R/W	Register No.
15	14	13	……	0		
			……		R/W	25 (hex)
X2IN (L)	X2FN					

X2FN: X2 Fractional Number

Bit Assignment							R/W	Register No.
15	14	13	12	11	……	0		
▨	▨	▨	▨		……		R/W	24 (hex)
———				X2IN (H)				

X2IN: X2 Integer Number

BEZIER Y2 [BY2] REGISTER

Bit Assignment					R/W	Register No.
15	14	13	……	0		
			……		R/W	2D (hex)
Y2IN (L)	Y2FN					

Y2FN: Y2 Fractional Number

Bit Assignment							R/W	Register No.
15	14	13	12	11	……	0		
▨	▨	▨	▨		……		R/W	2C (hex)
———				Y2IN (H)				

Y2IN: Y2 Integer Number

BEZIER X3 [BX3] REGISTER

Bit Assignment					R/W	Register No.
15	14	13	....	0		
			....		R/W	27 (hex)
X3IN (L)		X3FN				

X3FN: X3 Fractional Number

Bit Assignment							R/W	Register No.
15	14	13	12	11	....	0		
▨	▨	▨	▨		....		R/W	26 (hex)
———				X3IN (H)				

X3IN: X3 Integer Number

BEZIER Y3 [BY3] REGISTER

Bit Assignment					R/W	Register No.
15	14	13	....	0		
			....		R/W	2F (hex)
X3IN (L)		Y3FN				

Y3FN: Y3 Fractional Number

Bit Assignment							R/W	Register No.
15	14	13	12	11	....	0		
▨	▨	▨	▨		....		R/W	2E (hex)
———				Y3IN (H)				

Y3IN: Y3 Integer Number

(10) REGISTERS USED for DDA COMMANDS

6 registers and a counter are used for DDA commands.

The registers 0~2 (DDAR0~2) indicate variables, while the registers 3~5 (DDAR3~5) are set to constant values. The register 5 (DDAR5) is used for DDA2 commands to draw ellipses which are not parallel to the x – y axes (only when the DDAMD7 bit of the MOD2 register is "1").

The counter (DDAC) specifies the number of steps of DDA operation.

DDA REGISTER 0 [DDAR0]

Bit Assignment			R/W	Register No.
15		0		
			R/W	31 (hex)
ACC0L				

ACC0L: Accumulator 0 Low Data

Bit Assignment			R/W	Register No.
15		0		
			R/W	30 (hex)
ACC0H				

ACC0H: Accumulator 0 High Data

DDA REGISTER 1 [DDAR1]

Bit Assignment			R/W	Register No.
15		0		
			R/W	33 (hex)
ACC1L				

ACC1L: Accumulator 1 Low Data

Bit Assignment			R/W	Register No.
15		0		
			R/W	32 (hex)
ACC1H				

ACC1H: Accumulator 1 High Data

DDA REGISTER 2 [DDAR2]

Bit Assignment			R/W	Register No.
15	....	0		
	....		R/W	35 (hex)
ACC2L				

ACC2L: Accumulator 2 Low Data

Bit Assignment			R/W	Register No.
15	....	0		
	....		R/W	34 (hex)
ACC2H				

ACC2H: Accumulator 2 High Data

DDA REGISTER 3 [DDAR3]

Bit Assignment			R/W	Register No.
15	....	0		
	....		R/W	37 (hex)
CSTOL				

CSTOL: Constant 0 Low Data

Bit Assignment			R/W	Register No.
15	....	0		
	....		R/W	36 (hex)
CSTOH				

CSTOH: Constant 0 High Data

DDA REGISTER 4 [DDAR4]

Bit Assignment			R/W	Register No.
15		0		
CST1L			R/W	39 (hex)

CST1L: Constant 1 Low Data

Bit Assignment			R/W	Register No.
15		0		
CST1H			R/W	38 (hex)

CST1H: Constant 1 High Data

DDA REGISTER 5 [DDAR5]

Bit Assignment			R/W	Register No.
15		0		
CST2L			R/W	3B (hex)

CST2L: Constant 2 Low Data

Bit Assignment			R/W	Register No.
15		0		
CST2H			R/W	3A (hex)

CST2H: Constant 2 High Data

DDA COUNTER [DDAC]

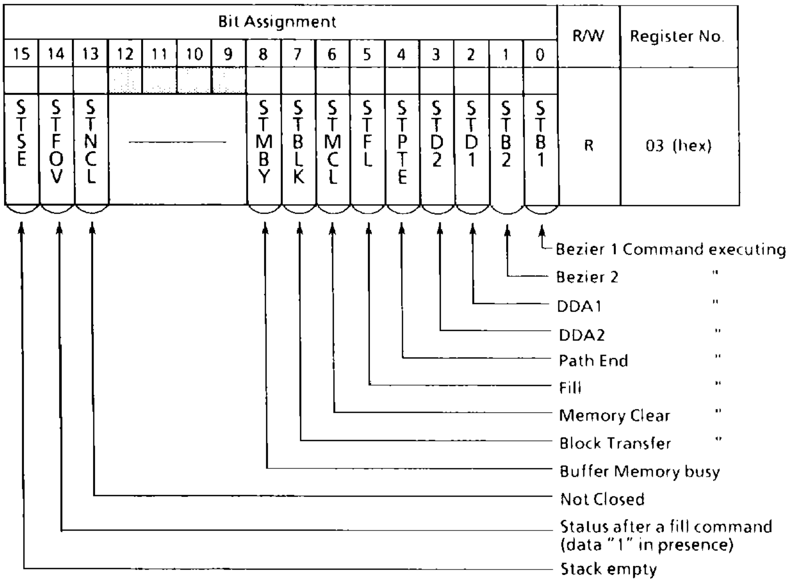
Bit Assignment			R/W	Register No.
15		0		
CUNT			R/W	3E (hex)

CUNT: Count Number

5.8.6 STATUS REGISTERS

The status registers indicate the internal states of the FGA. These are read-only registers.

(1) STATUS REGISTER 1 [SR1]



Bit Assignment																R/W	Register No.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
STSE	STFOV	STNCL	———				STMBY	STBLK	STMCL	STFL	STPTE	STD2	STD1	STB2	STB1	R	03 (hex)

Bezier 1 Command executing

Bezier 2 "

DDA1 "

DDA2 "

Path End "

Fill "

Memory Clear "

Block Transfer "

Buffer Memory busy

Not Closed

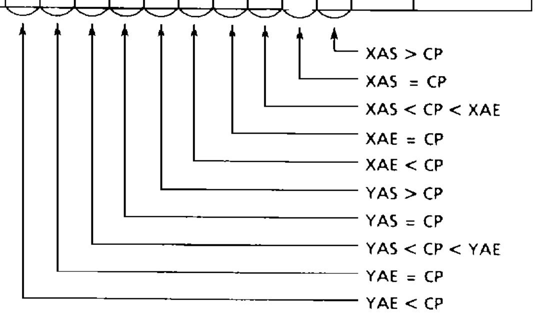
Status after a fill command (data "1" in presence)

Stack empty

(2) STATUS REGISTER 2 [SR2]

This register tells the relative position of the current address (indicated by the current pointer) in a work area defined by ASD and AED.

Bit Assignment																R/W	Register No.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
▨	▨	▨	▨	▨	▨												
———						STCL9	STCL8	STCL7	STCL6	STCL5	STCL4	STCL3	STCL2	STCL1	STCL0	R	02 (hex)



5.8.7 INTERRUPT REGISTERS

(1) INTERRUPT REGISTER [IR]

Bit Assignment																R/W	Register No.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
▨	▨	▨	▨	▨	▨	▨											
———							INTIL	INTPE	INTCP	INTAC	INTSO	INTMB	INTD	INTB2	INTB1	R	04 (hex)

Bit	Cause of Interrupt	Set	Reset
0	End of a Bezier 1/2 command (Bezier stack pointer = 0)	Upon termination of a Bezier 1/2 command	By the reading of this register by the CPU. By the reset operation (Hardware/Software reset)
1	Division point generated by a Bezier 2 command	Upon termination of a Bezier2 command that divided a Bezier curve	"
2	End of a DDA1/2 command (DDA counter = 0)	Upon termination of a DDA1/2 command	"
3	End of a CFIL, CMCL or CBLK command	Upon termination of a CFIL, CMCL or CBLK command	"
4	Overflow of the stack	By an overflow of the Bezier stack	"
5	Overflow of the current pointer	By an overflow of the current pointer	"
6	Overextension of the clipping area	By an overextension of the clipping area	"
7	Unclosed outline	Upon termination of a Path End command that failed to close an outline	"
8	Illegal operation	By an illegal operation	"

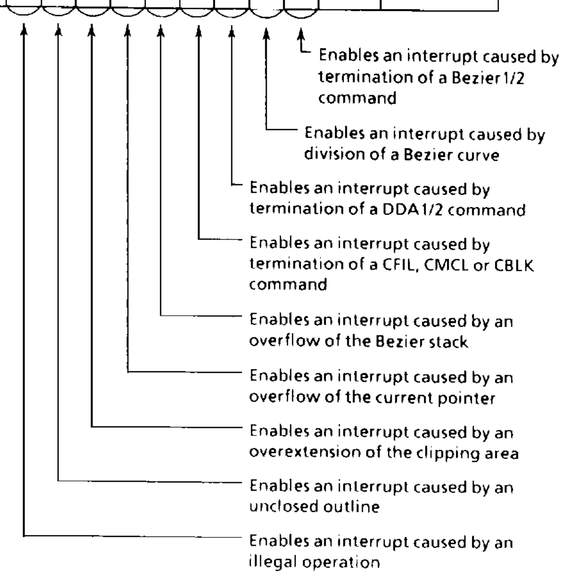
- Both the bits 0 and 1 are set upon termination of the final Bezier 2 command.
- All bits can be reset by the reading of the register, except those simultaneously set.
- An overflow of the Bezier stack should not occur if the system is operating normally.
- An overflow of the current pointer will not disturb the current operation.
- The beginning of an unclosed outline will be plotted in the buffer memory A, but not in the buffer memory B.
- The interrupt register must be reset upon occurrence of an interrupt which is not caused by termination of a command.

(2) INTERRUPT ENABLE REGISTER [IER]

The 9 types of interrupts are enabled or disabled by this register according to the status of the corresponding bits. Each bit can be masked or unmasked to enable interrupts on a selective basis.

An interrupt occurs if it is requested by the interrupt register and the corresponding bit is set to "1" on the interrupt enable register. The interrupt bits of the IR register can be set or reset independently from the interrupt enable bits.

Bit Assignment																R/W	Register No.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
▨	▨	▨	▨	▨	▨	▨											
—							IENIL	IENPE	IENCP	IENAC	IENSO	IENMB	IEND	IENB2	IENB1	R/W	05 (hex)



Enables an interrupt caused by termination of a Bezier 1/2 command

Enables an interrupt caused by division of a Bezier curve

Enables an interrupt caused by termination of a DDA 1/2 command

Enables an interrupt caused by termination of a CFIL, CMCL or CBLK command

Enables an interrupt caused by an overflow of the Bezier stack

Enables an interrupt caused by an overflow of the current pointer

Enables an interrupt caused by an overextension of the clipping area

Enables an interrupt caused by an unclosed outline

Enables an interrupt caused by an illegal operation

All bits are reset to 0

5.8.8 MEMORY PORT [MP] REGISTER

This register is used for the CPU to access the buffer memories (for random access, transfer of the results of fill commands and block transfer).

Bit Assignment			R/W	Register No.
15		0		
MPL			R/W	09 (hex)

MPL: Memory Port Low Data

Bit Assignment			R/W	Register No.
15		0		
MPH			R/W	08 (hex)

MPH: Memory Port High Data

5.8.9 DX, DY PORT [DXYP] REGISTER

This register is used for the CPU to read displacement data (dx, dy) generated by Bezier1, DDA1 and DDA2 commands, and to generate dx, dy directly to the buffer memories.

Bit Assignment							R/W	Register No.
15		4	3	2	1	0		
			DYS	DYO	DXS	DXO	R/W	0B (hex)

DXO: dx Absolute Bit
DXS: dx Sign Bit
DYO: dy Absolute Bit
DYS: dy Sign Bit

6. ELECTRICAL CHARACTERISTICS

6.1 ABSOLUTE MAXIMUM RATINGS

PARAMETER	SYMBOL	RATING	UNIT
Supply voltage	V_{DD}*	−0.5~+7.0	V
Input voltage	V_{IN}*	−0.3~+7.0	V
Operating temperature	T_{OPR}	0~+70	°C
Storage temperature	T_{STG}	−55~+150	°C

* If VSS = 0.

Caution) An operation outside the absolute maximum ratings may cause permanent damage to the device. For an expected reliability, the FGA should be used under the recommended operating conditions.

6.1.1 RECOMMENDED OPERATING CONDITIONS

PARAMETER	SYMBOL	MIN.	TYP.	MAX.	UNIT
Supply voltage	V_{DD}*	4.75	5	5.25	V
Input voltage	V_{IL}*	0	—	0.8	V
	V_{IH}*	2.2	—	VDD	V
Operating temperature	T_{OPR}	0	25	70	°C

* If VSS = 0.

6.2 DC CHARACTERISTICS

VDD = 5.0V ± 5% VSS = 0V

PARAMETER		SYMBOL	CONDITION	MIN.	MAX.	UNIT
Input High Voltage	CLOCK	V_{IH1}	(*1)	3.8	VDD	V
	Total input (except CLOCK)	V_{IH2}	(*2)	2.2	VDD	V
Input low voltage	CLOCK	V_{IL1}	(*1)	0	0.8	V
	Total input (except CLOCK)	V_{IL2}	(*2)	0	0.8	V
Input leak current	-CS, R/W, RA5~RA0, W/-LW,- DACK, CLOCK, -RESET	I_{IN}	V_{IN} = 0~VDD (*3)	– 10	10	μA
3-state input current (OFF state)	D31~D0, MDA7~MDA0, MDB7~MDB0	I_{TIN}	V_{IN} = 0.4~VDD (*4)	– 10	10	μA
Output high current	A24~A0, MDA7~MDA0, MDB7~MDB0	I_{OH1}	V_{OH} = VDD – 0.4 (*5)	—	– 0.5	mA
	D31~D0, -DTACK,-DREQ, -DSA, -DSB, MRA/-MWA, MRB/-MWB	I_{OH2}	V_{OH} = VDD – 0.4 (*6)	—	– 1.5	
Output low current	A24~A0, MDA7~MDA0, MDB7~MDB0	I_{OL1}	V_{OL} = 0.4 (*5)	1	—	mA
	D31~D0, -DTACK,-DREQ, -DSA, -DSB, MRA/-MWA, MRB/-MWB	I_{OL2}	V_{OL} = 0.4 (*6)	2	—	
	-IRQ	I_{OL3}	V_{OL} = 0.4 (*7)	4	—	
Output leak current (OFF state)	-IRQ	I_{LOH}	VOH = VDD (*7)	—	10	μA
Output capacity	-IRQ	C_{OUT}	(*7)	—	30	PF
Power consumption		I_{DD}	Reading/writing over the data bus Command execution		100	mA

*1 : CLOCK
*2 : All input terminal except CLOCK
*3 : -CS, R/W, RA5~RA0, W/LW, -DACK, CLOCK, -RESET
*4 : D31~D0, MDA7~MDA0, MDB7~MDB0
*5 : A24~A0, MDA7~MDA0, MDB7~MDB0
*6 : D31~D0, -DTACK, -DREQ, -DSA, -DSB, MRA/MWA, MRB/MWB
*7 : -IRQ

6.3 AC CHARACTERISTICS

VDD = 5.0V ± 5% VSS = 0V

PARAMETER	SYMBOL	Clock cycle 10MHz				UNIT
		MIN	MAX	MIN	MAX	
Clock Cycle Time	T_{CYC}	100	—			ns
Clock "Low" Level Width	T_{CLW}	43	—			ns
Clock "High" Level Width	T_{CHW}	43	—			ns
Clock Rise Time	T_{CLR}	—	10			ns
Clock Fall Time	T_{CLF}	—	10			ns
R/W Setup Time	T_{RWS}	30	—			ns
R/W Hold Time	T_{RWH}	10	—			ns
RAi Setup Time	T_{RAS}	30	—			ns
RAi Hold Time	T_{RAH}	10	—			ns
-CS Setup Time	T_{CSS}	40	—			ns
-CS Hold Time	T_{CSH}	30	—			ns
Read Data Access Time	T_{RAC}	—	(*1)			ns
Data Bus 3 State Recovery Time	T_{DBRT}	50	—			ns
Read Wait Time	T_{RWAT}	0	—			ns
Read Data Hold Time	T_{RDH}	—	40			ns
Read Data Turn Off Time	T_{RDZ}	—	70			ns
-DTACK Delay Time (Z to L)	T_{DTKZL}	—	40			ns
-DTACK Delay Time (D to L)	T_{DTKDL}	0	—			ns
-DTACK Turn Off Time (H to Z)	T_{DTKZ}	—	60			ns
-DTACK Hold TIme (L to H)	T_{DTKLH}	—	40			ns
Write Delay Time (-CS TO L)	T_{WDCL}	—	(*2)			ns
Write Wait Time	T_{WWAT}	0	—			ns
Write Data Setup Time	T_{WDS}	40	—			ns
Write Data Hold Time	T_{WDH}	30	—			ns
-DREQ Delay Time	T_{DRQD}	—	30			ns
DMA R/W Setup Time	T_{DRWS}	30	—			ns
DMA R/W Hold Time	T_{DRWH}	10	—			ns
-DACK Setup Time	T_{DAKS}	40	—			ns
-DACK Hold Time	T_{DAKH}	30	—			ns
DMA Read Data Access Time	T_{DRDAC}	—	(*3)			ns
DMA Read Wait TIme	T_{DRW}	0	—			ns
DMA Read Data Hold Time	T_{DRDH}	—	40			ns
DMA Read Data Turn Off Time	T_{DRDZ}	—	70			ns
DMA -DTACK Delay Time (Z to L)	T_{DDTZL}	—	40			ns

VDD = 5.0V ± 5% VSS = 0V

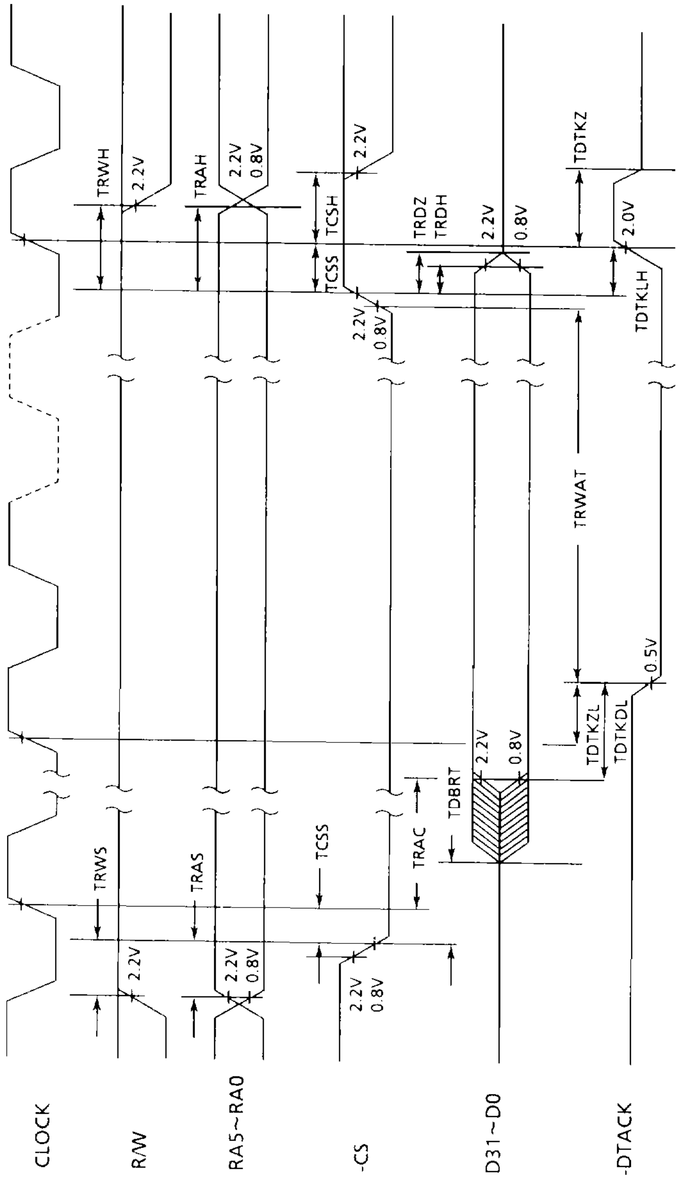
PARAMETER	SYMBOL	Clock cycle 10MHz				UNIT
		MIN	MAX	MIN	MAX	
DMA -DTACK Delay Time (D to L)	T_{DDTDL}	0	—			ns
DMA -DTACK Turn Off Time (H to Z)	T_{DDTHZ}	—	60			ns
DMA -DTACK Hold Time (L to H)	T_{DDTLH}	—	40			ns
DMA Write Delay Time (-DACK to L)	T_{DWDDL}	—	(*4)			ns
DMA Write Wait Time	T_{DWW}	0	—			ns
DMA Write Data Setup Time	T_{DWDS}	40	—			ns
DMA Write Data Hold Time	T_{DWDH}	30	—			ns
Memry Address Delay Time	T_{MAD}	—	40			ns
Memory Address Hold Time	T_{MAH}	—	30			ns
MRA (B)/MWA (B) Delay Time	T_{MRWD}	—	40			ns
MRA (B)/MWA (B) Hold Time	T_{MRWH}	—	30			ns
DSA (B) Delay Time	T_{DSD}	—	50			ns
DSA (B) Hold Time	T_{DSH}	—	30			ns
Memory Read Data Setup Time	T_{MRDS}	30	—			ns
Memory Read Data Hold Time	T_{MRDH}	20	—			ns
Memory Write Data Setup Time	T_{MWDD}	—	60			ns
Memory Write Data Hold Time	T_{MWDH}	—	30			ns
Memory Write Data Turn Off Time	T_{MWDZ}	—	40			ns
-RESET Input Pulse Width	T_{RES}	150	—			ns
-IRQ On Delay Time	T_{IRON}	—	100			ns
-IRQ Off Delay Time	T_{IROFF}	—	300			ns

Note) May change according to the bus width, memory wait mode (MMW = 1 or 0) or the buffer memory access condition.

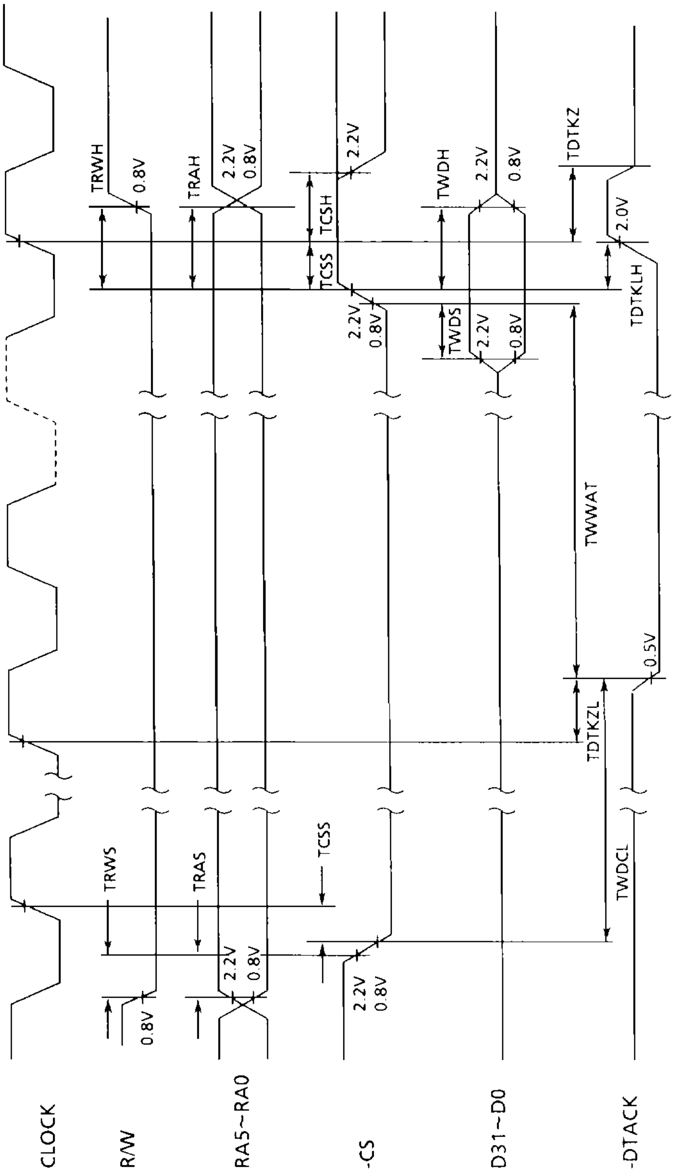
The worst value is following to.

*1	T_{RAC}	32-bit bus	(24 + 12W)	T_{CYC} + 50ns
		16-bit bus	(12 + 6W)	T_{CYC} + 50ns
*2	T_{WDCL}	32-bit bus	(7 + 4W)	$T_{CYC} + T_{DTKZL}$
		16-bit bus	(3 + 2W)	$T_{CYC} + T_{DTKZL}$
*3	T_{DRDAC}	32-bit bus	(15 + 8W)	T_{CYC} + 50ns
		16-bit bus	(7 + 4W)	T_{CYC} + 50ns
*4	T_{DWDDL}	32-bit bus	(7 + 4W)	$T_{CYC} + T_{DDTZL}$
		16-bit bus	(3 + 2W)	$T_{CYC} + T_{DDTZL}$

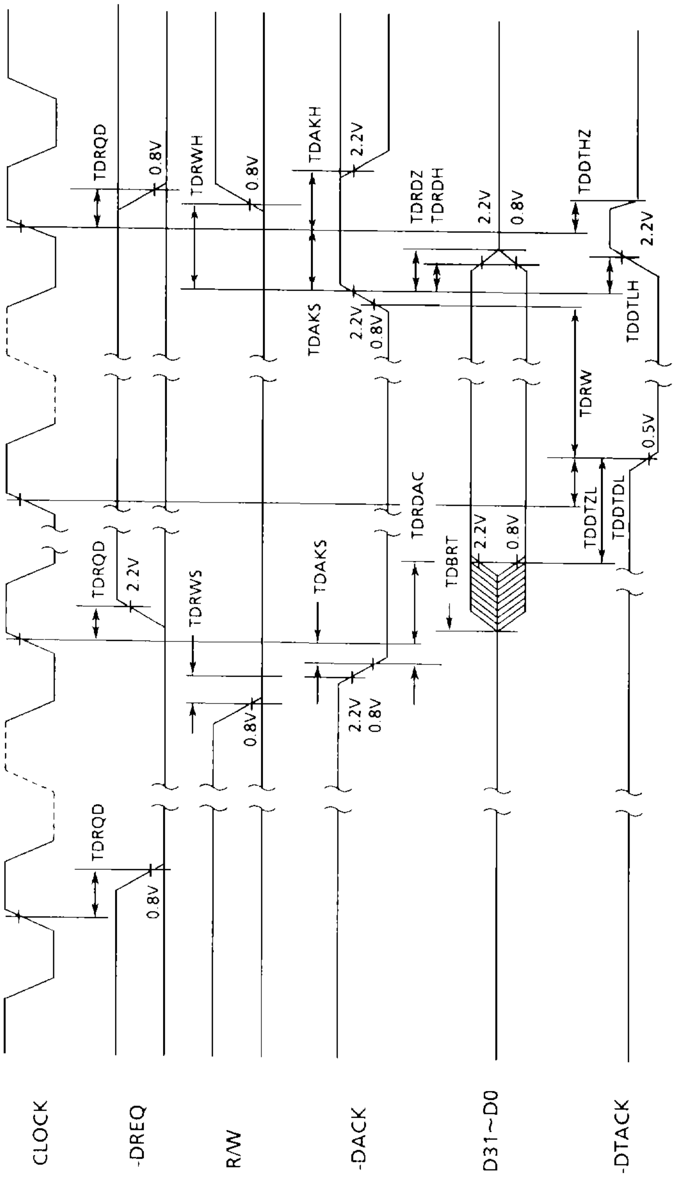
W : Number of memory wait (0~3).



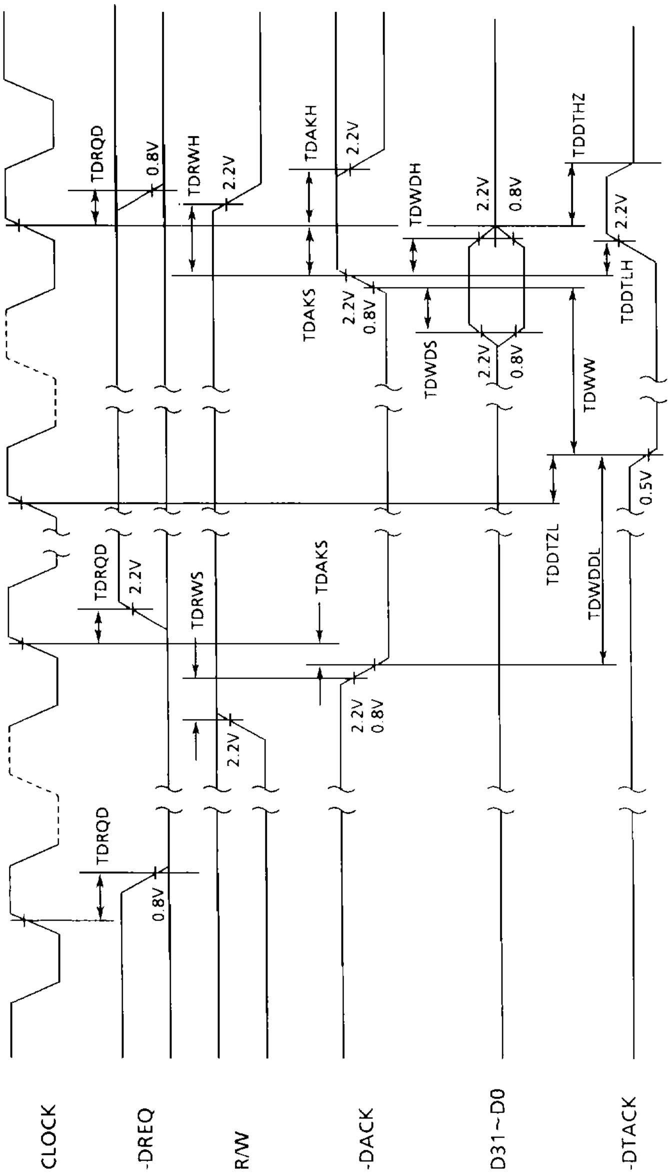
HOST READ CYCLE TIMING (HOST ← FGA)



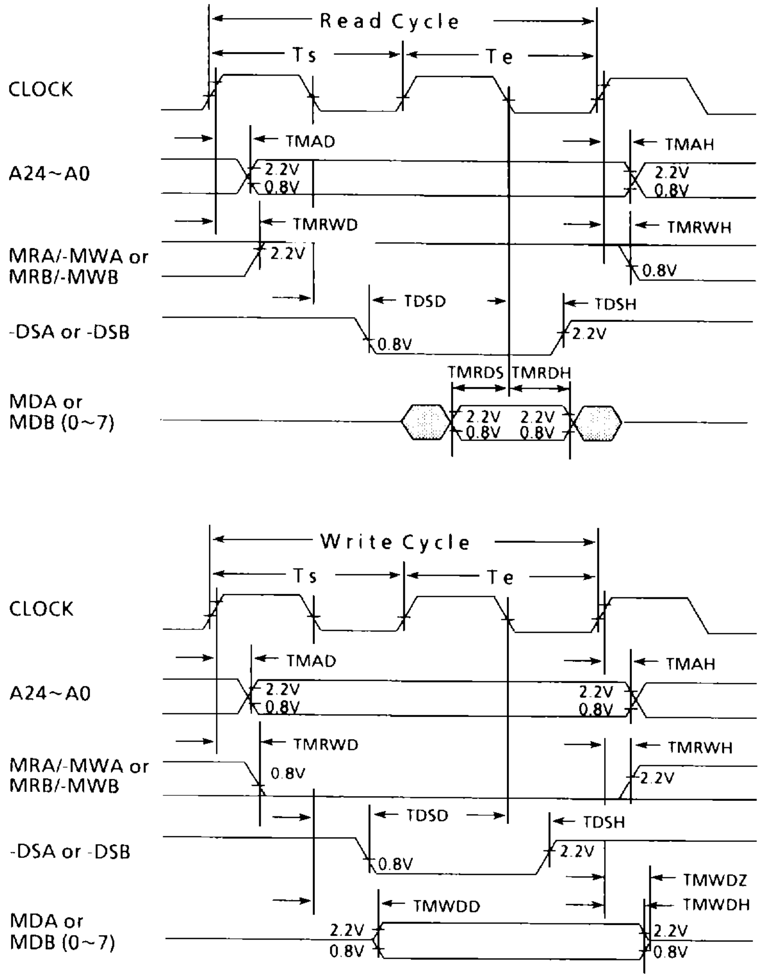
HOST WRITE CYCLE TIMING (HOST → FGA)



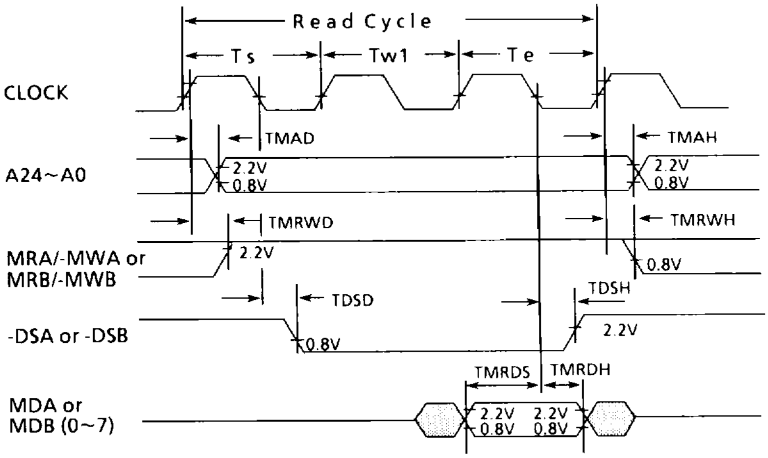
DMA READ CYCLE TIMING (HOST MEMORY ← FGA)

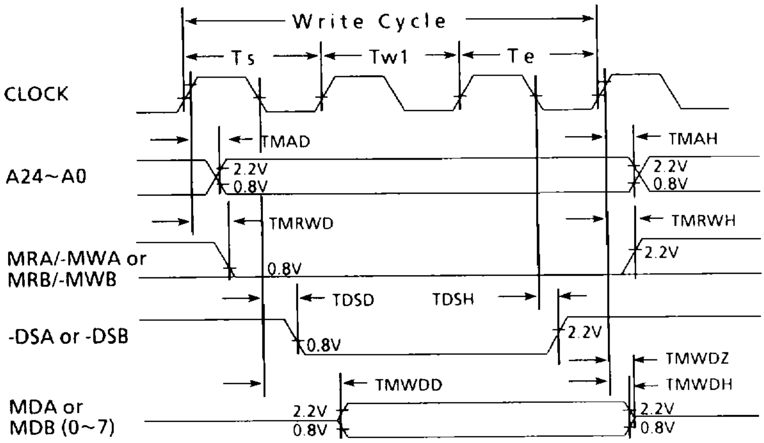


DMA WRITE CYCLE TIMING (HOST MEMORY — FGA)



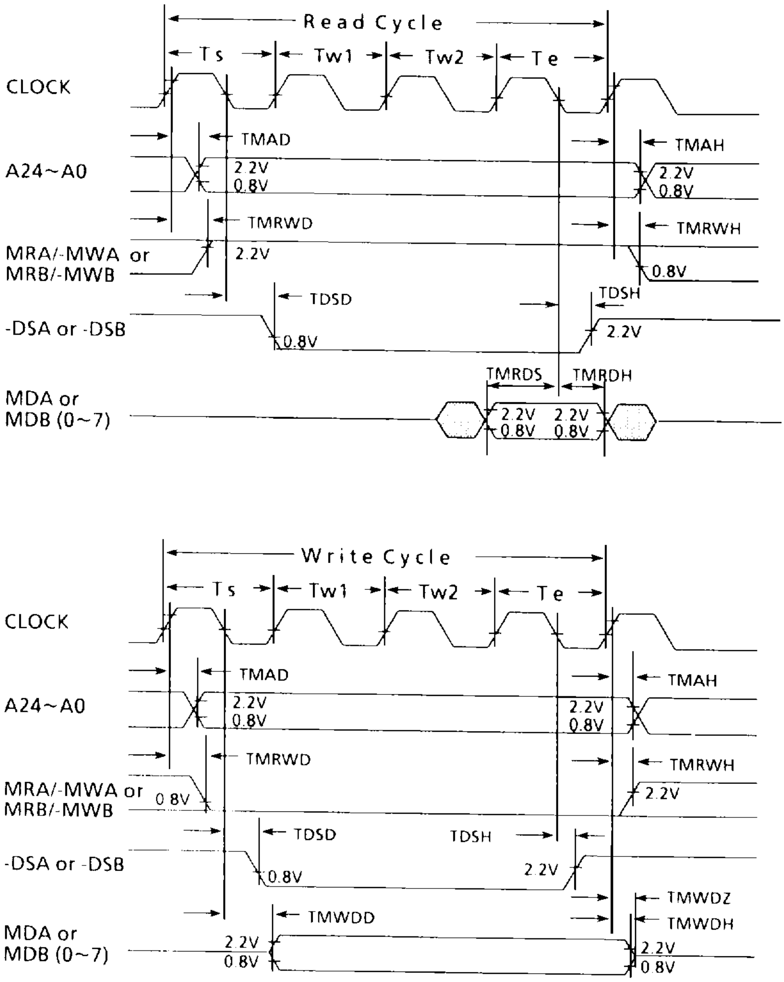
MEMORY CYCLE TIMING (NO WAIT)





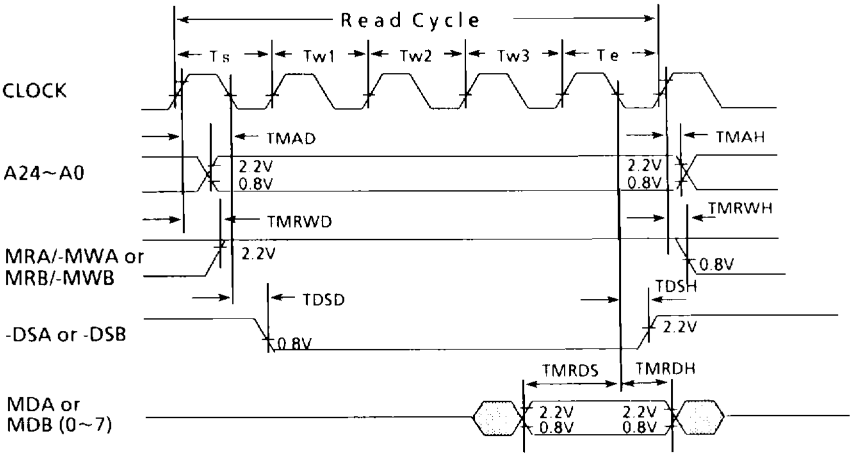
Corresponding Memory Tacc 100~ 120ns

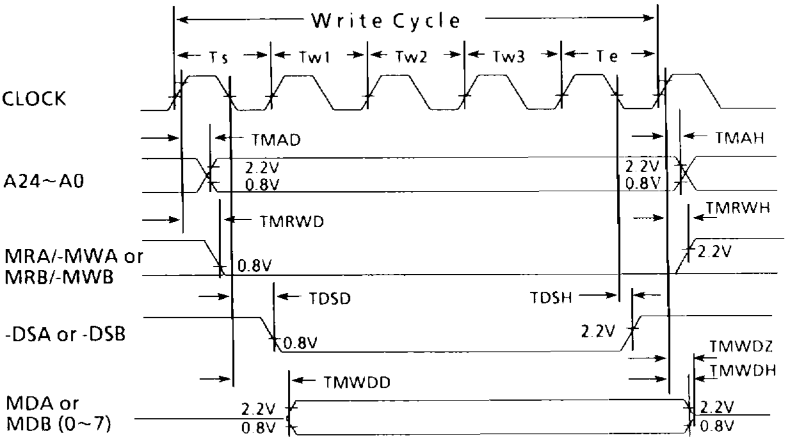
MEMORY CYCLE TIMING (1 WAIT)



Corresponding Memory Tacc 150 ns

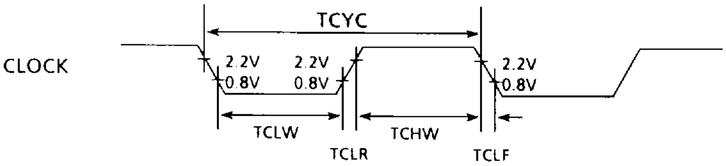
MEMORY CYCLE TIMING (2 WAIT)



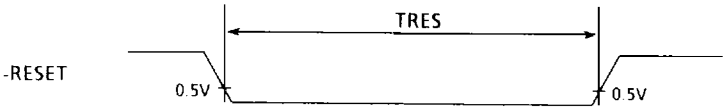


Corresponding Memory Tacc 200ns

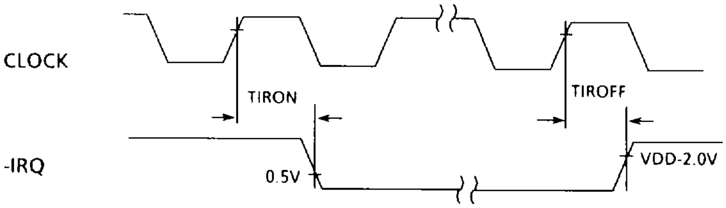
MEMORY CYCLE TIMING (3 WAIT)



TIMING OF CLOCK



TIMING OF -RESET INPUT

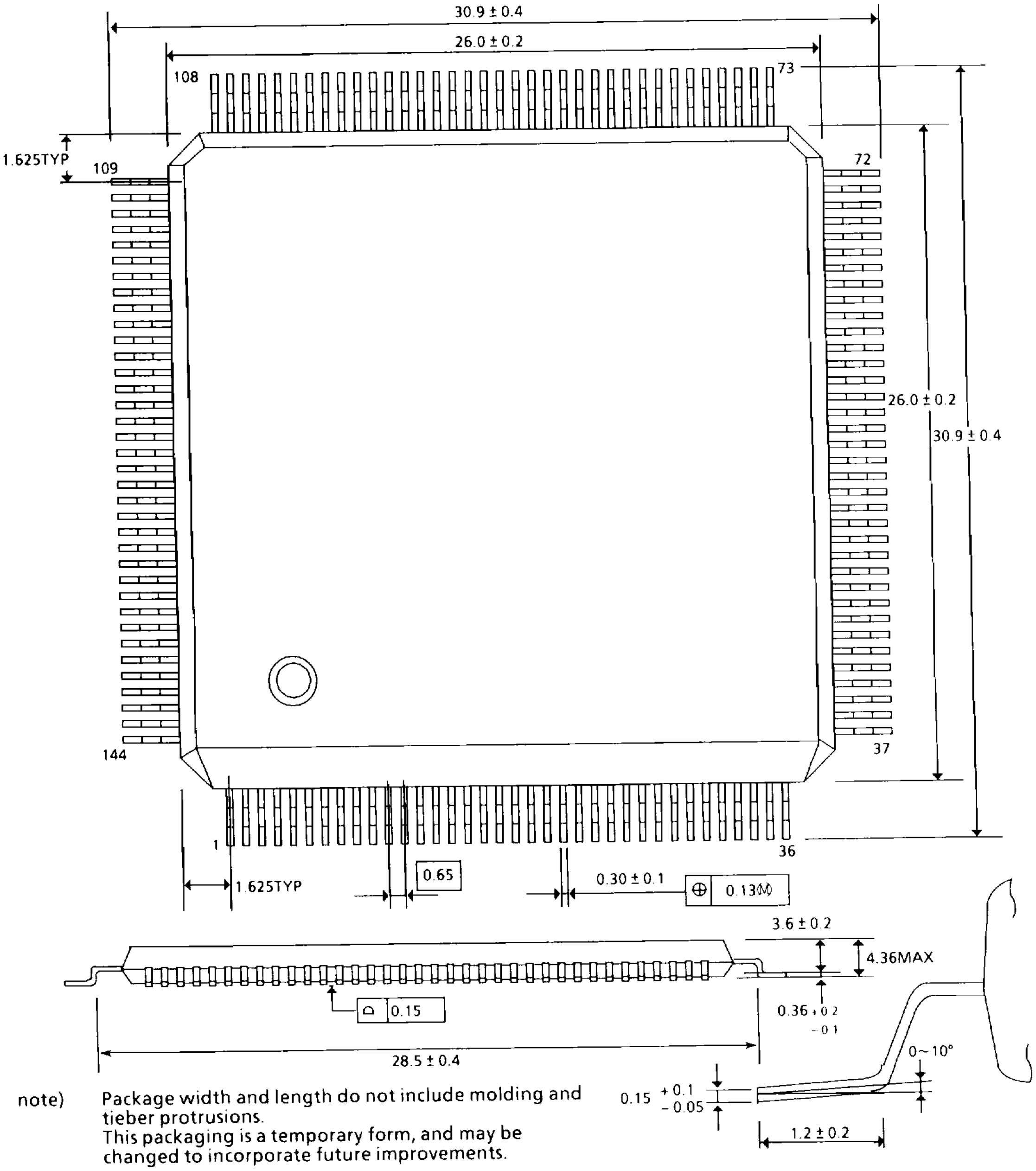


TIMING OF -IRQ OUTPUT

7. PACKAGE DIMENSION

QFP144-P-2626B

UNIT : mm



note) Package width and length do not include molding and tieber protrusions.
This packaging is a temporary form, and may be changed to incorporate future improvements.