CHAPTER 5

TOKEN DEFINITIONS

The microcode is accessed by sending tokens and data parameters down the FIFO. The microcode gets the token and uses its associated data parameter values to perform the desired function. If the function is expected to return any data the results are placed in known locations of the microcode data RAM. These locations are read by the host software to get the return data. The following paragraphs define the tokens available in the MGR version of the GE5 microcode. Each token description has the following sections:

- token description
- data parameter definitions
- return values in the GE5 data RAM
- example usage

The example usage is intended to show the token and data parameter type and order of usage.It is not intended to necessarily show an exact real world usage. However the examples are intended to show the steps the host software must take to property interface with the microcode for each token.

The tokens are listed in alphabetical order.

GE_ABNORMAL

This token is used to set the ABNORMAL flag in microcode data RAM to the specified value. The ABNORMAL flag indicates if the normal is to be renormalized.The host sets this flag whenever the normal matrix stack is changed.

Data Parameters :

data type	parameter name	parameter description
int	abflag	indicates if renormaiiiation is needed or not. -1 renormalization is not needed 1 renormaiiiation is needed

Return **Data from Microcode Data RAM :**

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     abfiag = 1;     /* indicate renormalization needed */

GEWAIT;
GE[GE_ABNORMAL].i = 0;
GE[GE_DATA].i = abflag;
```

GE_ANTIALIAS

This token is used to enable or disable antialiasing of lines. If the data parameter is 0 then antialiasing is turned off. If the data parameter is not 0 then antialiasing is turned on.

Data Parameters :

data type	parameter name	parameter description
int	aliasflag	sets the **anitialiase** mode or disables **antialiasing** 0 - antialiasing off 3 - **4 bit** double buffer CI mode 9 - 8 bit single buffer CI mode 9 - 12 bit double buffer CI mode

Return Data from **Microcode** Data RAM **:**

None

Example Usage **:**

```
#include ":ng:.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

Int     aliasflag = 0; /* turn off antialiasing */

GEWAIT;
GE[GE_ANTIALIAS].i = 0;
GE[GE_DATA].i = aliasflag;
```

GE_AUXWRITEMASK

This token is used to set the Raster Engine AUXMASK register. This register contains 9 bits which are used as a write mask for the PUP, UAUX, WID and Z buffer bitpfanes. A 0 bit disables writes and a 1 bit enables writes. The base configuration has only two auxiliary bitplanes and two window ID bitplanes. In this configuration bits 7-6 and 3-2 have no meaning and should be set to zero. The PUP, UAUX or WID bitplanes to be written to are selected with the GE_DRAWMODE token. Since these bitplanes can only be selected individually the GE_AUXWRITEMASK should be set appropriately for the biilanes being accessed. The following settings allow the various bitplanes to be accessed without affecting the other bitplanes.

0x003	to write to just the PUP bitplanes
oxooc	to write to just the UAUX bitplanes
0x0F0	to write to just the WID bitplanes
0x100	to write to just the Z Suffer bitplanes

Data Parameters :

data type	parameter name	parameter description
int	auxmask	auxiliary write mask value
	bits	bitplanes masked
	1 - o	PUP bitplanes
	3 - 2	UAUX bitplanes
	7 - 4	WI D bitplanes
	6	Z buffer bitplanes

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     auxmask = 0x10; /* set mask for WID bitplane 0 */

GEWAIT;
GE[GE_AUXWRITEMASK].i = 0;
GE[GE_DATA].i = auxmask;
```

GE_BACKFACE

This token is used to enable and disable **backface** polygon removal. A **backface** polygon has it's vertices defined in a clockwise order.

Data Parameters :

data type	parameter name	parameter description
int	**bf_flag**	**backface** polygon removal flag 1 enable **backface** poly removal -1 disable **backface** poly removal

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     bf_flag = 1;

GEWAIT;
GE[GE_BACKFACE].i = 0;
GE[GE_DATA].I = bf_flag;
```

GE_BEGINBBOX

This token is used to begin bounding box mode. Two culling parameters specify the x and y size for the smallest displayable feature. The bounding box is then specified using the GE-POLYGON, GE_VERTEX3 and GE_ENDPOLYGON tokens.

Data Parameters :

data type	parameter name	parameter description
int	minx	minimum x size in pixels
Int	miny	minimum y size in pixels

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     minx = 10, miny = 20;

GEWAIT;
GE[GE_BEGINBBOX].i = 0;
GE[GE_DATA].i = minx:
GE[GE_DATA].i = miny;
```

GE_BEGINMESH

This to&en is used to mark the start of a list of vertex routines interpreted as points in a triangle mesh.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_BEGINMESH].i = 0;
```

GE_CLOSEDLINE

This token is used to mark the start of closed line mode which draws line segments between the specified vertices.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_CLOSEDLINE].i = 0;
```

GE_CMOV2

This token is used to move the current character position to the 2-D location specified by the absolute world coordinate floating point data parameters. The next **GE_DRAWCHAR** token sent will display it's character at the specified x, y location.

Data Parameters :

data type	parameter name	parameter description
float	cx	character x bcatbn
float	cy	character y location

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float    cx = 4.0, cy = 2.0;

GEWAIT;
GE[GE_CMOV2].i = 0;
GE[GE_DATA].f = cx;
GE[GE_DATA].f = cy;
```

GE_CMOV2I

This token is used to move the current character position to the 2-D location specified by the absolute world coordinate integer data parameters. The **next GE_DRAWCHAR** token sent will display it's character at the specified x, y location.

Data Parameters :

data type	parameter name	parameter description
int	cx	character x location
int	cy	character y location

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

Im_GEsetup;

int     cx = 4, cy = 2;

GEWAIT;
GE[GE_CMOV2I].i = 0;
GE[GE_DATA].i = cx;
GE[GE_DATA].i = cy;
```

GE_CMOV3

This token is used to move the current character position to the 3-D location specified by the absolute world coordinate floating point data parameters. The next **GE_DRAWCHAR** token sent will display it's character at the specified x, **y, z** location.

Data Parameters :

data type	parameter name	parameter description
float	cx	character x bcatbn
float	**cy**	character y bcatbn
float	a	character **z** location

Return Data from Mlcrocode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float    cx = 4.0, cy = 2.0, a = 6.0;

GEWAIT;
GE[GE_CMOV3].i = 0;
GE[GE_DATA].f = cx;
GE[GE_DATA].f = cy;
GE[GE_DATA].f = cz;
```

GE_CMOV3I

This token is used to move the current character position to the 3-D location specified by the absolute world coordinate integer data parameters. The next GE_DRAWCHAR token sent will display it's character at the specified x, **y, z** location.

Data Parameters :

data type	parameter name	parameter description
int	cx	character x location
int	**cy**	**character y** location
int	**a**	character **z** location

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h

Im_GEsetup;

int     cx = 4, cy = 2, cz = 6;

GEWAIT;
GE[GE_CMOV3I].i = 0;
GE[GE_DATA].i = cx;
GE[GE_DATA].i = cy;
GE[GE_DATA].i = a;
```

GE_CMOV4

This token is used to move the current character position to the 3-D location specified by the absolute homogeneous coordinate floating point data parameters. The next GE_DRAWCHAR token sent will display it's character at the specified x, y, z, w location.

Data Parameters :

data type	parameter name	parameter description
float	cx	character x location
float	cy	character y location
float	cz	character z location
float	cw	character w location

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float    cx = 4.0, cy = 2.0, cz = 6.0, cw = 1.0;

GEWAIT;
GE[GE_CMOV4].i = 0;
GE[GE_DATA].f = cx:
GE[GE_DATA].f = cy;
GE[GE_DATA].f = cz;
GE[GE_DATA].f = cw;
```

GE_CMOV4I

This token is used to move the current character position to the 3-D location specified by the absolute homogeneous coordinate integer data parameters. The next GE_DRAWCHARtoken sent will display it's character at the specified x, **y, z, w** bcatipn.

Data Parameters :

data type	parameter name	parameter description
int	cx	character x location
int	**cy**	character y **location**
int	a	character **z** location
int	cw	character w **location**

Return Data from Microcode Data RAM **:**

None

Example **Usage :**

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     cx = 4, cy = 2, cz = 6, cw = 1;

GEWAIT;
GE[GE_CMOV4I].i = 0;
GE[GE_DATA].i = cx;
GE[GE_DATA].i = cy;
GE[GE_DATA].i = a;
GE[GE_DATA].i = cw;
```

GE_COLOR

This token is used to set a color index value using an integer data parameter. The **GE_DRAWMODE** token determines which bitplanes the color value is written to. It can be be written to the Frame Buffer biplane for color index pixels (see GE_PIXTYPE) or It can be written to the PUP, UAUX or WID bitplanes.

Data Parameters :

data type	parameter name	parameter description
int	color	**color** index color value for drawing 0 - **0xFF** for base configuration 0 - **0xFFF** for enhanced configuration

Return Data from Mlcrocode Data RAM **:**

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     color = 5;    /* Magenta */

GEWAIT;
GE[GE_COLOR].i = 0;
GE[GE_DATA].i = color;
```

GE_COLORF

This token is used to set a color index value with the floating point color index. The GE_DRAWMODE token determines which bitplanes the color value is written to. It can be be written to the Frame Buffer bitplane for color index pixels (see GE_PIXTYPE) or it can be written to the PUP, UAUX or WID bitplanes.

Data Parameters :

data type	parameter name	parameter description
float	colorf	color index color value for drawing

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float colorf = 1.0; /* Red */

GEWAIT;
GE[GE_COLOR].i = 0;
GE[GE_DATA].f = colorf;
```

GE_COMPOSEMATRIX

This token is used to **concatenate** two matrices whii are **located** on the special **PHIGSs** matrix stack. The **PHIGSs** stack occupies the bottom six matrix entries of the sixteen total matrix entries. This allows the matrix concatenation *to be* done at the Geometry Engine speeds rather than at the host speeds which are considerably slower. The data parameters specify the matrix index of the two matrices which are to be concatenated and the index of the matrix in which the result is to be stored. This token is not currently used by **any** of the graphics library calls.

Data Parameters :

data type	parameter name	parameter description
int	src_index1	source matrix index 1
int	**src_index2**	source matrix index 2
int	result-index	result matrix index

Return Data from **Microcode Data RAM :**

None

Example Usage **:**

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

Int     src_index1 = 11, src_index2 = 12;
int     result-index = 13;

GEWAIT;
GE[GE_COMPOSEMATRIX].i = 0;
GE[GE_DATA].i = src_index1;
GE[GE_DATA].i = src_index2;
GE[GE_DATA].i = result-index;
```

GE-CONCAVE

This token is used to enable and disable concave polygon mode. When concave polygon mode is enabled the microcode expects concave polygons.

Data Parameters :

data type	parameter name	parameter description
Int	cp_flag	concave polygon flag 1 concave polygon mode enabled -1 concave polygon mode disabled

Return **Data from Microcode Data RAM :**

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     cp_flag = 1;

GEWAIT;
GE[GE_CONCAVE].i = 0;
GE[GE_DATA].i = cp_flags;
```

GE_COPYMATRIX

This token is used to copy a matrix from one location to another on the special PHIGSs matrix stack. The PHIGSs stack occupies the bottom six matrix entries of the sixteen total matrix entries. The data parameters specify the source matrix index which is to be copied and the destination matrix index where the copy will be stored. This token is not currently used by any of the graphics library calls.

Data Parameters :

data type	parameter name	parameter description
int	src_index	source matrix index
int	dest_index	destination matrix index

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     src_index = 13;
int     dest_index = 15;

GEWAIT;
GE[GE_COPYMATRIX].i = 0;
GE[GE_DATA].i = src_index;
GE[GE_DATA].i = dest_index;
```

GE_CTX0

This token is used to initiate a graphics context switch in the microcode. It has a value of 0xF0 which causes the HQ1 to recognize the token as a context switch token. Before this token is sent the finish 1 flag must be cleared because it will be set when the microcode is ready to do a DMA of the current context or of the new context.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "gecmds.h"

long        zero = 0

FINISH_WR(FINISH1_OFF, 0); /* clear finish flag 1 */

FIFO_WR(GE_CTX0, zero):
```

GE_CTX1

This token is used to continue the start of a graphics context switch. It has a value of 0xFF which causes the HQ1 to recognize it as a context switch token. This token is sent immediately after the GE_CTXO token. This is to handle the case where the microcode was in the process of doing two FIFO read operations when the GE_CTX0 token was sent. The GE_CTX1 token prevents the second FIFO read from stalling the microcode forever (or until the kernel timed out and did a reset).

This token is followed by two data parameters. The first parameter indicates if the current context needs to be saved or not. If it does then it has to be DMAed out of the microcode data ram to the host. The second parameter indicates the DMA s&e of the new incoming context.

Data Parameters :

data type	parameter name	parameter description
float	old-state	the state of the current context -1.0 - CTX_DEAD (can be overwritten) 1.0 - CTX_ALIVE (needs saving)
float	new-site	the DMA size of the incoming context -1.0 - CTX_SMALL 1.0 - CTX_BIG 2.0 - CTX_NEW

Return Data from Microcode **Data RAM :**

Address	Description
DMA_COUNT	Contains the DMA word count for the context to be removed
'PCSAVE	Contains the microinstruction where the restored context will resume
● PCSAVE + 1 (+2)	Next microinstruction

Example Usage :

```
#include "mgr.h"
#include "gecmds.h"
#include "ge5_glob.h"

long    zero = 0;

long    context-state = CTX_ALIVE;

FIFO_WR(GE_CTX1, zero):

FIFO_WR(GE_DATA, CTX_ALIVE);
FIFO_WR(GE_DATA, CTX_SMALL);

for (i-0 ;i< TiMEOUT_CXRDY ;i++){
    if (FiNiSH_POLL(FiNiSHI_OFF))
        break:
    /* Don't poll every time through the loop */
    DELAY(P);    /* a small 2 microsecond delay */
```

```
}

/* if a timeout occurred then the MGR adapter must have some problem or the microcode is
stalled. Can report the error or initiate a full adapter reset and reinitialiration */

if (i == TIMEOUT_CXRDY)
    /* do a return with an error code so the error can be handled */

/* if the current context is alive and must be saved */

if (context-state -- CTX_ALIVE ) {
    FINISH_RD(FINISH0_OFF, curcxp->finish0);   /* save finish 0 flag in host context area */
    HQM_WR(DMA_COUNT);                         /* set address to the DMA_COUNT address */
    DRAM_RD(DMA_COUNT, count);                 /* get the DMA word count of the context to save */

    /* set up Eddy chip for microcode data RAM to host context save buffer DMA transfer */

}

if (new-context-state -- CTX_SMALL || new-context-state == CTX_BIG) {
    FINISH_WR(FINISH0_OFF, newcxp->finish0); /* restore finish 0 flag of incoming ctx */

    /* Do the ctx rollback operation to adjust the microcode to account for context switching in
    the middle of a token parameter list */

    HQMMSB_WR(1);                  /* set HQ mid address MSB reg to access ucode RAM */

    HQRDPC(savepc);                /* save the current GE5 PC */

    /* get the saved pc from the outgoing context which is still in host memory */

    pc = *(long *)(gfxcxp + GR1_GETOCX(PCSAVE)) & 0x7FFF;      /* low 15 bits only */

    HQM_WR(pc);          /* set the HQMAR to the saved pc address */

    URAM_RD(pc, inst); /* read the microcode instruction at that address */

    /* check the microcode instruction and the following one for a memptr autoIncrement, an
    reptr autoincrement. If either type is found then decrement the MEMPTR or the REPTR field
    in the context to be loaded */

    if (inst & CX_INCMEM) {
        *(gfxcxp + GR1_GETOCX(MEMPTR)) -- 1;   /* decrement memptr to be restored */

    if (inst & CX_INCRE) {
        *(gfxcxp + GR1_GETOCX(REPTR)) -= 1;    /* decrement memptr to be restored */

    if (inst & CX_FIELD2) {
        pc +=2;   /* adjust pc by 2 for double word instruction */
    else
        pc++;     /* adjust pc by 1 for single word instruction */

    HQM_WR(pc);
    URAM_RD(pc, inst);          /* read the microcode instruction */
```

```
    if (inst 8 CX_INCMEM) {
        '(gfxcxp + GR1_GETOCX(MEMPTR)) -= 1;   /* decrement memptr to be restored */

    if (inst & CX_INCRE) {
        '(gfxcxp + GR1_GETOCX(REPTR)) -- 1;   /* decrement memptr to be restored ● /

    /* restore the GE5 PC saved above so the context switch microcode can continue */

    HQM_WR((long)savepc);

    URAM_RD((long)savepc, inst); /* reading URAM resets the PC to the specified address ● /

    /* now get the HQ MAR register previously saved value and restore it ● /

    HQM_WR(*(long ● )(gfxcxp + GR1_GETOCX(HQMSAV));

    HQMMSB_WR(0);   /* reset MAR MSB reg to 0 ● /

    /* The outgoing context's rollback operation is now complete */

    /* Set up the second Eddy DMA channel for the host to ucode data ram DMA transfer */

}

FINISH_WR(FINISH1_OFF, 0):          I' clear finish flag 1 ● I

/* Unstall the GE5 so it can do the DMA operation */

HQMMSB_WR(1);       /* set MSB reg to 1 to issue clear stall command */
HQCLRSTL();         /* issue the clear stall command ● /
HQMMSB_WR(0);       /* reset the MSB reg */

/* Start the ucode data RAM to host buffer DMA transfer on channel 0. Channel 1 DMA will
continue as soon as channel one DMA completes*/

for (i=0; i < TIMEOUT_CXDONE; i++) {
    if (FINISH_POLL(FINISH1_OFF))
            break:
    /* Don't poll every time through the loop */
    DELAY(2);   /* a small 2 microsecond delay */
}

/* if a timeout occurred then the MGR adapter must have some problem or the microcode is not
happy about something.  Can report the error or initiate a full adapter reset and reinitialization
* /

if ( i == TIMEOUT_CXDONE)
    /* do a return with an error code so the error can be handled */

/* The old context is saved in the host memory and the outgoing context has been loaded and
should be active as far as the adapter is concerned */
```

GE_CURRENTWID

This token is used to set the CURWID register in the Raster Engine. The CURWID register is used to hold the current window ID (WID) used for window **ID checking.** When window ID checking is enabled, a pixel write to the frame buffer occurs only when the current window ID matches the window ID bitplanes for the selected pixel. **For** the base configuration the current WID Is only 2 bits. For the enhanced configuration the current WID is 4 bits. If the fast **Z** clear mode is enabled (DEPTHFN bit 3 set) then only the upper 3 bits of the current WID **are used for the** comparison.

Data Parameters :

data type	parameter name	parameter description
int	currentwid	specified current Window **ID**

Return Data from Microcode Data RAM :

None

Example Usage :

```
- #include "mgr.h"
  #include "imsetup.h"
  #include "gecmds.h"

  im_GEsetup;

  Int     currentwid = 1;     /* Window ID equal 1 */

  GEWAIT;
  GE[GE_CURRENTWID].i = 0;
  GE[GE_DATA].i = currentwid;
```

GE_CURVEIT

This token is used to perform fast curve rendering by specifying an iteration count. The iteration count is used to iterate the matrix on the top of the matrix stack as a forward difference matrix. This token is preceded by a GE_MOVE3I token and its coordinate data. The curve is then drawn by performing the matrix iteration and drawn the appropriate line segments.

Data Parameters :

data type	parameter name	parameter description
float	itcount	the matrix iteration count

Return **Data from Microcode Data RAM :**

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float       itcount = 10.0;

GEWAIT;
GE[GE_CURVEIT].i = 0;
GE[GE_DATA].f = itcount;
```

GE_CZCLEAR

This token is used to clear the image bitplanes and the Zbuffer bitplanes. The image bitplanes are cleared to the specified color. The color is either an **RGB** color or a color index depending on the pixel type specified by the **GE_PIXTYPE** token. The **fastzflag** indicates if the fast **z** dear mode should be used to clear the **Z** Buffer. If **fastzclear** is set to 1 then a fast Z buffer clear is done by setting bit 3 of GE_DEPTHFN and by setting the least significant bit of the window ID bits. The fast Z clear mode can only be used on the enhanced adapter. Refer to the Raster Subsystem chapter for additional details on the fast **z** clear method of clearing the Z buffer bitplanes. If the faskclear bit is -1 then the Z buffer is actually cleared to the value in the **zvalue** parameter. If a raster operation other than 3 has been specified by the GE_RASTEROP token then the faskclear flag is ignored and the Z Buffer is cleared using the slower method of writing the **zvalue** to the Z Buffer bitplanes.

If **PICKMODE** is active the bitplanes will not be cleared and the **MOREHITS** flag will be set. If the SIMPLE flag is set to 1 then the fast block write clear will be performed unless the faskciear flag is set for a slow clear. The SIMPLE flag should be set to 1 if the window is 1 to 4 pieces and it should be set to -1 if the window is **five** or more pieces. The **czclear** microcode clears **only** the rectangles described by the piece list set using the GE_SETPIECES token. Also the **RE2** hardware screen mask set with the GE_SCRMASK token clips the pixel writes which are outside the screen mask rectangle. Refer to the GE_SCREENCLEAR token for additional information.

Data Parameters :

data type	parameter name	parameter description
int	**redcolor**	Red color value or Cl value
int	greencolor	Green color value or 0 if Cl mode
int	bluecolor	Blue color value or 0 if Cl mode
int	**zvalue**	Z Value to be loaded into Z buffer
int	faskflag	Fast Zbuffer clear flag -1 • disable fast zciear 1 • enable fast zciear

Return Data from Microcode Data RAM **:**

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     redcolor = 0xFF, greencolor = 0, bluecolor = 0x80; /* some strange color */
int     zvalue = 0;    /* z-value isn't used if doing a fast z clear */
int     faskflag = 1; /* Enable fast z clear (only on the enhanced adapter) */

GEWAIT;
GE[GE_CZCLEAR].i = 0;
GE[GE_DATA].i = redcolor;
```

```
GE[GE_DATA].i = greencolor;
GE[GE_DATA].i = bluecolor;
GE[GE_DATA].i = zvalue;
GE[GE_DATA].i = fastzflag;
```

GE-DATA

This token is not a command token but is instead used to send parameter data for the other tokens. The data can be either a 32 bit integer or an IEEE format 32 bit floating point value. The use of this token is shown in the other token descriptions which have data parameters. The value written can be a constant or a variable.

Example Usage :

```
/* An integer constant, refer to the various token descriptions for examples of integer variable usage */

GE[GE_DATA].i = 1;        /* integer parameter */

/* A floating point constant, refer to the various token descriptions for examples of floating point variable usage */

GE[GE_DATA].f = 2.0;      /* floating point parameter */

/* If floating point data cannot be sent down the FIFO a work around example */

float     temp;

temp = 2.0;

GE[GE_DATA].i = *(int*)&temp;
```

GE_DEPTHCUE

This token is used to enable or disable depth cueing. When depth cueing is enabled the shaderange or RGB shaderange value along with the **z** value determine the pixels color.

Data Parameters :

data type	parameter name	parameter description
int	**dc_flag**	**depth** cue mod8 flag 0 - disable depth cue mode 1 - enable depth cue mode

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
tIncludo "gcomds.h"

im_GEsetup;

int         dc_flag = 1;

GEWAIT;
GE[GE_DEPTHCUE].i = 0;
GE[GE_DATA].i = dc_flag;
```

GE_DEPTHFN

This token is used to specify the comparison function for the **Z** comparison circuitry in the RE2.. This token causes the DEPTHFN register in the RE2 to be immediately changed. Normally the **GE_ZFUNCTION** token specifies the depth function used during **Z** Buffer mode and this token would be used to temporarily override the current **Z** function value while doing a **Z** Buffer clear.

The depth function is composed of four bits with the first three specifying the relational comparison function and the fourth bit controlling the enabling or disabling of the fast Z clear mode of operation. If bit 3 is set and the least significant bit in the Window ID is set then the source value is made invalid and the compare passes. The function specified would cause the Z compare to always pass and would enable the fast Z dear for the enhanced system and would disable it for the base system.

Data Parameters **:**

data type	parameter name	parameter description
int	depthfn	desired Z depth comparison function 0x7 for base system so Z compare always passes and fast Z dear is disabled **0xF** for enhanced system **so** Z compare always passes and fast Z clear is enabled

Return **Data from Microcode Data RAM :**

None

Example **Usage :**

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     depthfn = 0xF;      /* sat for fast Z clear on enhanced system */

GEWAIT;
GE[GE_DEPTHFN].i = 0;
GE[GE_DATA].i = depthfn;
```

GE_DRAW2

This token is used to draw a line from the current graphics drawing position to the specified 2-D position. The destination position is specified in absolute world coordinates and the coordinate data parameters are specified using floating point format. After the line is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
float	mx	absolute x graphics position
float	my	absolute y graphics position

Return Data from Microcode **Data RAM :**

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float    mx = 4.0, my = 2.0;

GEWAIT;
GE[GE_DRAW2].i = 0;
GE[GE_DATA].f = mx;
GE[GE_DATA].f = my:
```

GE_DRAW2I

This token is used to draw **a line from the** current graphics drawing position to the specified 2-D position. The destination position is spedfied in absolute world coordinates and the coordinate data parameters **are** specified using integer format. After the line is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
Int	mx	absolute x graphics position
int	my	absolute y graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     mx = 4, my = 2;

GEWAIT;
GE[GE_DRAW2I].i = 0;
GE[GE_DATA].i = mx;
GE[GE_DATA].i = my;
```

GE_DRAW3

This token is used to draw a line from the current graphics drawing position to the specified 3-D position. The destination Position is specified in absolute world coordinates and the coordinate data parameters are specified using floating point format. After the line is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
float	mx	absolute x graphics position
float	my	absolute y graphics position
float	mz	absolute z graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#Include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float    mx = 4.0, my = 2.0, mz = 6.0;

GEWAIT;
GE[GE_DRAW3].i = 0;
GE[GE_DATA].f = mx;
GE[GE_DATA].f = my;
GE[GE_DATA].f = mz;
```

GE_DRAW3I

This token is used to draw a line from the current graphics drawing position to the specified 3-D position. The destination position is specified in absolute world coordinates and the coordinate data parameters are specified using integer format. After the line is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
int	mx	absolute x graphics position
int	my	absolute y graphics position
int	mz	absolute z graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     mx = 4, my = 2, mz = 6;

GEWAIT;
GE[GE_DRAW3I].i = 0;
GE[GE_DATA].i = mx;
GE[GE_DATA].i = my;
GE[GE_DATA].i = mz;
```

GE_DRAW4

This token is used to draw a line from the current graphics drawing position to the specified 3-D position. The destination position Is specified in absolute homogeneous coordinates and the coordinate data parameters are specified using floating point format. After the line is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
float	mx	absolute x graphics position
float	my	absolute y graphics position
float	mz	absolute z graphics position
float	mw	absolute w graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float   mx = 4.0, my = 2.0, mz = 6.0, mw = 1.0:

GEWAIT;
GE[GE_DRAW4].i = 0;
GE[GE_DATA].f = mx;
GE[GE_DATA].f = my;
GE[GE_DATA].f = mz;
GE[GE_DATA].f = mw;
```

GE_DRAW4I

This token is used to draw a line from the current graphics drawing position to the specified 3-D position. The destination position is specified in absolute homogeneous coordinates and the coordinate data parameters are specified using integer format. After the line is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
int	mx	absolute x graphics position
int	my	absolute y graphics position
int	mz	absolute z graphics position
int	mw	absolute w graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     mx = 4, my = 2, mz = 6, mw = 1;

GEWAIT;
GE[GE_DRAW4I].i = 0;
GE[GE_DATA].i = mx;
GE[GE_DATA].i = my;
GE[GE_DATA].i = mz:
GE[GE_DATA].i = mw;
```

GE_DRAWCHAR

This token is used to draw a character bitmap on the screen at the current character position **(cpos).** After the character is drawn the **cpos** is updated to the next character position. The data parameters include the character metrics as shown in Figure 5.1. After the character metrics are sent the character bitmap data is sent.

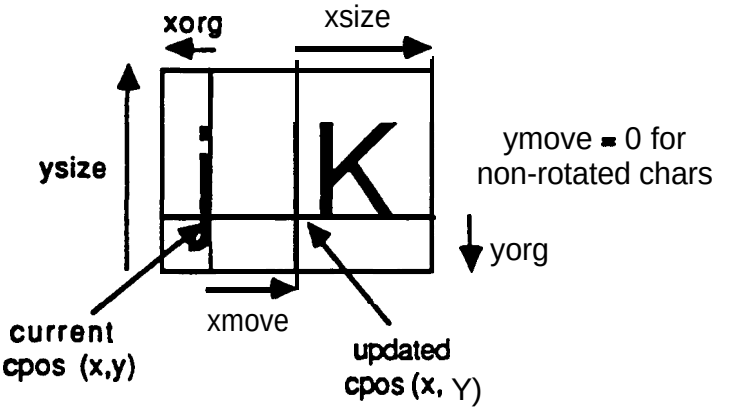

Figure 5.1 Character Metrics

Dara Parameters :

data type	parameter name	parameter description
int	xsize	character bitmap x dimension in bits
int	ysite	character bitmap y dimension in bits
int	sper	specifies the number of **bitmask** values per row
int	xorig	x offset size in bits from current char position to left side of bitmap
int	yorig	y offset size in bits from current char position to bottom of bitmap
int	xmove	distance in bits to be added to **cpos** x value for new **cpos** x value
int	ymove	distance in bits to be added to **cpos** y value for new **cpos** y value
int	**bitmask**	**bitmask** data (sper X ysire)

Return Data from Microcode Data RAM

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     xsize = 16, ysize = 16, sper = 1;
int     xorig = 0, yorig = 4;
```

```
int         xmove = 16, ymove = 0;
int         i, bitmask_size = sper * ysize;
short bitmask[16] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}; /* space */

GEWAIT;
GE[GE_DRAWCHAR].i = 0;
GE[GE_DATA].i = xsize;
GE[GE_DATA].i = ysize;
GE[GE_DATA].i = sper;
GE[GE_DATA].i = xorig;
GE[GE_DATA].i = yorig;
GE[GE_DATA].i = xmove;
GE[GE_DATA].i = ymove;
for (i = 0 ; i < bitmask_size ; i++)
    GE[GE_DATA].i = bitmask[i];
```

GE_DRAWMODE

This token is used to set the drawing mode. This mode determines which bitplanes will be accessed for drawing operations. Three data-parameters are sent down to the micro&de which are used as branch flags for setting the appropriate microcode and Raster Engine drawing mode parameters. This token does not affect the Z buffer bitplanes, refer to the GE_ZFUNCTION token to enable or disable drawing into the Z Buffer bitplanes.

Data Parameters :

data type	parameter name	parameter description
int	brflagl	microcode branch flag 1
int	brflag2	microcode branch flag 2
int	brflag3	microcode branch flag 3

brflag 1	brflag2	brflag3	drawing mode
1	1	1	NORMALDRAW, FBUFFERDRAW, ZBUFFERDRAW mode destination bitplanes are Frame Buffer bitplanes
-1	1	1	WIDDRAW mode destination bitplanes are WID bitplanes
-1	-1	1	PUPDRAW destination bitplanes are PUP bitplanes
-1	-1	-1	OVERDRAW or UNDERDRAW destination bitplanes are UAUX bitplanes

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     brflagl = 1, brflag2 = 1, brflag3 = 1; /* NORMALDRAW */

GEWAIT;
GE[GE_DRAWMODE].i = 0;
GE[GE_DATA].i = brflag1;
GE[GE_DATA].i = brflag2;
GE[GE_DATA].i = brflag3;
```

GE_ENABDITH

This token is used to enable or disable dithering.

Data Parameters :

data type	parameter name	parameter description
int	dflag	dither flag enable/disable value 0 - disable dithering 1 - enable dithering

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     dflag = 1;

GEWAIT;
GE[GE_ENABDITH].i = 0;
GE[GE_DATA].i = dflag;          /* Enable dithering */
```

GE_ENABLWID

This token is used to enable window ID checking for lines. For rectangular windows which are unobscured (1 piece) the Geometry Engine clips lines to the window boundary using the screen mask values so window ID checking for lines should be disabled. For other window shapes or obscured windows (more than 1 piece) the window ID checking for lines should be enabled. When window ID checking is enabled, pixels can only be written that have a window ID the same as the current window ID set by **GE_CURRENTWID.** The window manager normally changes the ENABLWID value in the context and when the context is restored the ENABLWID value takes affect. This token is not normally used to enable line wid checking since it is a window manager responsibility.

Data Parameters :

data type	parameter name	parameter description
int	**l_wid_flag**	window ID checking for lines flag **0 - disable** WID checking for lines 1 - enable WID checking for lines

Return Data from Microcode Data RAM **:**

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#Include "gecmds.h"

im_GEsetup;

int     l_wid_flag = 1;

GEWAIT;
GE[GE_ENABLWID].i = 0;
GE[GE_DATA].i = l_wid_flag;
```

GE_ENABWID

This token is used to enable window ID checking. When window ID checking is enabled the current WID (set by GE_CURRENTWID) is compared to the WID bitplane for the current pixel. If they are the same then the new pixel value is written to the frame buffer. The FLATMODE and FLATDX values in the data RAM would need to be set also. The window manager normally changes these valuesin the context and when the context is restored the FLATMODE and FLATDX values take affect. The FLATMODE variable would be set to 1 and the FLAT DX variable would be set to 0x4000. This only affects polygon drawing and rectangular drawing which uses shaded spans such as the sbox and rectangle drawing tokens. The GE_ENABLWID token is used to enable fine drawing wid checking.

Data Parameters :

data type	parameter name	parameter description
int	wid_flag	window ID checking flag 0 - disable WID checking 1 - enable WID checking

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     wid_flag = 1:

GEWAIT;
GE[GE_ENABWID].i = 0;
GE[GE_DATA].i = wid_flag;
```

GE_ENDBBOX

This token is used to end the bounding box mode.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_ENDBBOX].i = 0;
```

GE_ENDCLOSEDLINE

This token is used to end closed line mode. A line segment is drawn between the final vertex and the first vertex which was specified after the **GE_BGNCLOSEDLINE** token. The current graphics position becomes undefined.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_ENDCLOSEDLINE].i = 0;
```

GE_ENDFEEDBACK

This token is used to end feedback mode. The microcode will cause a GE interrupt when the feedback data buffer is full. The microcode will generate a final GE interrupt when this token is received so the host can read the partial feedback data buffer contents. The address and size of the data will be stored in microcode data RAM locations. When the GE interrupt occurs the feedback data is read from microcode data RAM and is placed in a host buffer.

Data Parameters :

None

Return Data from Microcode **Data RAM :**

Address	Description
PICKPTR	address pointer to picking or feedback data
PICKSPACE	size of the picking or feedback data

Example Usage :

```
#include "mgr.h"
#include "gecmds.h"
#include "ge5_glob.h"

/* DATA-SIZE would need to be defined to be large enough to hold the feedback data */

long        addr, size, buffer[DATA_SIZE];
register  long  i;

FIFO_WR(GE_ENDFEEDBACK,  0);

/* This code is assumed to be part of the interrupt service routine for the GE interrupt */

/* GE interrupt handler processes the GE interrupt and reads the feedback data from microcode
RAM. The interrupt handler would have to save the HQ MAR register since it gets changed. */

HQM_WR(PICKPTR);
DRAM_RD(PICKPTR,  a d d r );/*;feedback data address */

HQM_WR(PICKSPACE);
DRAM_RD(PICKSPACE, size):     /* word count */

/* Transfer the feedback data from microcode data RAM to a host buffer*/

HQM_WR(addr);

for (i = 0 ; i < size ; i++) {
    DRAM_RD(addr++,  buffer++);
    if ((addr % HQM_PG_SIZE) == 0)
        HQM_WR(addr);
}
```

```
/* Clear GE interrupt in HQ and on EDDY module */
/* Restore the HQ MAR register and return from interrupt handier */
```

GE_ENDMESH

This token is used to stop interpreting vertices as triangle mesh vertices.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_ENDMESH].i = 0;
```

GE_ENDOLDPOLYGON

This token is used to indicate that all of the vertices and any other necessary information have been sent and the previously specified drawing operation can be performed. The drawing operations include the fill polygon operation, the drawa shaded polygon operation or the draw a filled rectangle operation.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_ENDOLDPOLYGON].i = 0;
```

GE_ENDPICKMODE

This token is used to end pick mode. The microcode will cause a GE interrupt when the pick data buffer is full. The address and size of the data will be stored in microcode data RAM locations. When the GE interrupt occurs the pick data is read from microcode data RAM and is placed in a host buffer. When this token is received by the microcode one final GE interrupt is generated so the partial pick buffer can be read. The method of reading the pick mode data from the microcode data RAM is the same as for reading the feedback data.

Data Parameters :

None

Return Data from Microcode Data RAM :

Address	Description
PICKPTR	address pointer to picking or feedback data
PICKSPACE	size of the picking or feedback data

Example Usage :

```
#include "mgr.h"
#include "gecmds.h"
#include "ge5_glob.h"

/* DATA-SIZE would need to be defined to be large enough to hold the pick data */

long    addr, size, buffer[DATA_SIZE];
register long i;

FIFO_WR(GE_ENDPICKMODE, 0);

/* This code is assumed to be part of the interrupt service routine for the GE interrupt */

/* The GE interrupt handler processes the GE Interrupt and reads the pick data from microcode
 RAM. The Interrupt handler would have to save the HQ MAR register since it gets changed. */

HQM_WR(PICKPTR);
DRAM_RD(PICKPTR, addr);      /* pick data address */

HQM_WR(PICKSPACE);
DRAM_RD(PICKSPACE, size):    /* word count */

/* Transfer the pick data from microcode data RAM to a host buffer */

HQM_WR(addr);
for (i = 0 ; i < size ;i++){
    DRAM_RD(addr++, buffer++);
    if ((addr % HQM_PG_SIZE) == 0)
        HQM_WR(addr);
}
```

```
/* Clear GE interrupt In HQ and on EDDY module */
/* Restore the HO MAR register and return from interrupt handler */
```

GE_ENDPOLYGON

This token is used to indicate that the scope of GE-POLYGON has ended and that all the vertices of the current polygon have been received. The current graphics position is undefined after the microcode receives this token.

Data Parameters

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h

im_GEsetup;

GEWAIT;
GE[GE_ENDPOLYGON].i = 0;
```

GE_FATPOLY

This token is used enable or disable an old style of polygon drawing. This mode when enabled causes the polygon edge lines to be slightly wider.

Data Parameters :

data type	parameter name	parameter description
int	fp-flag	fat polygon mode flag 0 - disable fat polygon mode 1 - enable fat polygon mode

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

Int    fp_flag = 0;

GEWAIT;
GE[GE_FATPOLY].i = 0;
GE[GE_DATA].i = fp_flag;
```

GE_FDRAW

This token is used to perform a fast 2-D line draw operation from the current graphics position to the specified x, y coordinates. This token is not currently used by any of the graphics library calls.

Data Parameters :

data type	parameter name	parameter description
float	px	x graphics coordinate
float	py	y graphics coordinate

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float          px = 100, py =300;

GEWAIT;
GE[GE_FDRAW].i = 0;
GE[GE_DATA].f = px;
GE[GE_DATA].f = py;
```

GE-FEEDBACK

This token is used to turn on feedback mode.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_FEEDBACK].i = 0;
```

GE_FINISH0

This token is used to tell the microcode to set the FIFO empty finish flag. When the microcode encounters this token it sets finish flag 0 to indicate that the FIFO is empty.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h

im_GEsetup;

GEWAIT;
GE[GE_FINISH0].l = 0;
```

GE_FLATMODE

This token is used by the window manager to set the flat mode used when a polygon is filled. If the GE_SHADEMODEL token selects shaded fill mode then the **flatmode** value is ignored. If the shaded fill mode is flat then the **flatmode** determines whether the **RE2** shaded span instruction is used to draw a flat span or whether the flat span instruction is used. If WID checking is required then the window manager should set the **flatmode** to 1 so that the shaded span instruction is used since the shaded span instructions are WID checked. If WID checking is not needed then the window manager should set the **flatmode** to 2 so that the flat spans are used which are not WID checked and are faster than the shaded ***span*** instructions. The flatdx parameter specifies the RE2 DX parameter. If **flatmode** is set to 1 then the flatdx value should be set to 0x4000 (DX = +1). If **flatmode** is set to 2 then the flatdx should be set to 0 (DX = 0).

Data Parameters :

data type	parameter name	parameter description
Int	**flatmode**	specifies the polygon flat fill mode 1 - if WID checking enabled 2 - If WID checking disabled
int	flatdx	specifies the RE2 DX parameter 0x4000 if **flatmode** = 1 0 if **flatmode** = 2

Return Data from Microcode Data RAM **:**

None

Example Usage **:**

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     flatmode = 1;       /* WID checking required */
int     flatdx = 0x4000;    /* DX = +1 */

GEWAIT;
GE[GE_FLATMODE].i = 0;
GE[GE_DATA].i = flatmode;
GE[GE_DATA].i = flatdx;
```

GE_FLINE

This token is used to perform a fast 2-D line draw operation from the first set of x, y coordinates to the second set of x, y coordinates. This token is not currently used by any of the graphics library calls.

Data Parameters :

data type	parameter name	parameter description
float	**px1**	starting x graphics coordinate
float	**py1**	starting y graphics coordinate
float	**px2**	ending x graphics **wordinat8**
float	PY2	**ending y graphics coordinate**

Return Data **from Microcode Data RAM :**

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float          px1 = 100, py1 =300, px2 = 300, py2 = 600;

GEWAIT;
GE[GE_FLINE].i = 0;
GE[GE_DATA].f = pxl ;
GE[GE_DATA].f = py1;
GE[GE_DATA].f = px2;
GE[GE_DATA].f = py2;
```

GE_FMOVE

This token is used to set a new graphics drawing position to the specified x, y coordinates for fast 2D line draw operations. This token is not currently used by any of the graphics library calls.

Data Parameters :

data type	parameter name	parameter description
float	px	x graphics coordinate
float	PY	y graphics coordinate

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#Include "gecmds.h"

im_GEsetup;

float          px = 100, py =300;

GEWAIT;
GE[GE_FMOVE].i = 0;
GE[GE_DATA].f = px;
GE[GE_DATA].f = py;
```

GE_GETCPOS

This token is used to get the current character position x, y values. The microcode places the x location value in the microcode data RAM at location CPOSX. It places the y location value at microcode data RAM location CPOSX. If the x, y location is not valid a negative value is written at microcode location CHARVALID. If the location is valid then CHARVALID will be greater than or equal to zero.

Data Parameters :

None

Return Data from Microcode Data RAM

Address	Description
CPOSX	Current Character Position X value
CPOSY	Current Character Position Y value
CHARVALID	Valid position flag (< 0 means invalid)

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"

im_GEsetup;

long        xpos, ypos, valid;

GEWAIT;
GE[GE_GETCPOS].i = 0;

/* We have to make sure the microcode is stalled before reading the microcode data RAM for the
position. This is achieved by sending the GE_FINISH0 token and polling finish flag 0 until it is
set. */

FINISH_WR(FINISH0_OFF, 0);  /* clear finish flag 0 */
GE[GE_FINISH0].i = 0; /* Tell microcode to set finish 0 */
while (1) {
    if (FINISH_POLL(FINISH0_OFF)
            break;
}

/* Now read the character position. The HQ MAR is set to the desired address by using the
GL_HQM_WR macro. This macro sets the MAR register and also puts the MAR address in the
context save area in microcode data RAM in case a context switch should occur during the
process of reading the character position */

GL_HQM_WR(CPOSX);   /* set CPOSX address */
DRAM_RD(CPOSX, xpos);       /* read x position */
```

```
GL_HQM_WR(CPOSY);   /* set CPOSY address */
DRAM_RD(CPOSY, ypos);          /* read y position */
GL_HQM_WR(CHARVALID); /* set CHARVALID address */
DRAM_RD(CHARVALID, valid)  /* read char valid flag */
```

GE_HQMSAV

This token is used to instruct the microcode to store the specified page address in the HQMSAV location in the microcode data ram. The HQ MAR register is then set to the address by the host using the HQM_WR macro. The HQ MAR address is saved in the microcode data ram so that the host can restore the address in the event that the current context is switched out of the adapter and another context changes the HQ MAR register. This is necessary because the HQ MAR is write only from the host. This token is used as part of the GL_HQM_WR macro to save the new HQ MAR address as it is being set to a new value by a user level program. This also allows an interrupt routine to change the HQ MAR value as it desires and then restore the value to the previously saved value from the HQMSAV data ram location.

Data Parameters :

data type	parameter name	parameter description
int	address	new address for the HQ MAR

Return Data from Microcode Data RAM :

Address	Description
HQMSAV	Contains contents of HQ MAR register

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"

im_GEsetup;

int    address = PICKPTR; /* PICKPTR is just an example it could be any valid address */

GEWAIT;
GE[GE_HQMSAV].i = 0;
GE[GE_DATA].i = address;
```

GE_INIT

The GE_INIT token is used to instruct the microcode to perform the necessary operations to initialize the Geometry Engine and the Raster Engine.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_INIT].i = 0;
```

GE_INITNAMES

This token is used to initialize the name stack used for picking and selecting.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_INITNAMES].i = 0;
```

GE_LIGHTATTR1

This token is used to load lighting vector parameters. The data parameters specify the address in microcode data RAM where the other three lighting vector parameters will be stored. Only one of the parameters will contain lighting data and the other two will be zero.

Data Parameters :

data type	parameter name	parameter description
int	address	address in microcode data RAM
float	x	x portion of lighting vector
float	y	y portion of lighting vector
float	z	z portion of lighting vector

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

im_GEsetup;

int     address = K_M;      /* location in LIGHTDATA */
float   x = k; /* k is a lighting parameter */
float   y = 0.0, z = 0.0;

GEWAIT;
GE[GE_LIGHTATTR1].i = 0;
GE[GE_DATA].i = address;
GE[GE_DATA].f = x;
GE[GE_DATA].f = y;
GE[GE_DATA].f = z;
```

GE_LIGHTATTR2

This token is used to load lighting vector parameters. The data parameters specify the address in microcode data RAM where the other three lighting vector parameters will be stored. Only two of the parameters will contain lighting data and the third will be zero.

Data Parameters :

data type	parameter name	parameter description
int	address	address in microcode data RAM
float	x	x portion of lighting vector
float	y	y portion of lighting vector
float	z	z portion of lighting vector

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

im_GEsetup;

int     address = ATTENUATION_M; /* loc in LIGHTDATA */
float   x = att.x;      /* att.x is a lighting parameter */
float   y = att.y;      /* att.y is a lighting parameter */
float   z = 0.0;

GEWAIT;
GE[GE_LIGHTATTR2].i = 0;
GE[GE_DATA].i = address;
GE[GE_DATA].f = x;
GE[GE_DATA].f = y;
GE[GE_DATA].f = z;
```

GE_LIGHTATTR3

This token is used to load lighting parameters. The data parameters specify the address in microcode data RAM where the other three lighting parameters will be stored. All three of the parameters will contain lighting data.

Data Parameters :

data type	parameter name	parameter description
int	address	address in microcode data RAM
float	x	x portion of lighting vector
float	y	y portion of lighting vector
float	z	z portion of lighting vector

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

im_GEsetup;

int     address = CIS_M;    /* loc in LIGHTDATA */
float   x = ci.x;           /* ci.x is a lighting parameter */
float   y = ci.y;           /* ci.y is a lighting parameter */
float   z = ci.z;           /* ci.z is a lighting parameter */

GEWAIT;
GE[GE_LIGHTATTR3].i = 0;
GE[GE_DATA].i = address;
GE[GE_DATA].f = x;
GE[GE_DATA].f = y;
GE[GE_DATA].f = z;
```

GE_LIGHTDIRECTION

This token is used to load lighting direction data. The four data parameters contain the lighting direction which will be stored at the location in microcode data RAM previously specified by the GE_LIGHTMEMPTR token.

Data Parameters :

data type	parameter name	parameter description
float	x	x portion of lighting direction
float	y	y portion of lighting direction
float	z	z portion of lighting direction
float	w	w portion of lighting direction

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h"
#include "lighting.h"

im_GEsetup;

float    x = dirx;    /* lighting direction x */
float    y = diry;    /* lighting direction y */
float    z = dirz;    /* lighting direction z */
float    w = 1.0;     /* 1.0 for local lighting direction */

GEWAIT;
GE[GE_LIGHTDIRECTION].i = 0;
GE[GE_DATA].f = x;
GE[GE_DATA].f = y;
GE[GE_DATA].f = z;
GE[GE_DATA].f = w;
```

GE_LIGHTDATA4

This token is used to load lighting data. The four data parameters contain the lighting data which will be stored at the location in microcode data RAM previously specified by the GE_LIGHTMEMPTR token.

Data Parameters :

data type	parameter name	parameter description
float	x	x portion of lighting data
float	y	y portion of lighting data
float	z	z portion of lighting data
float	w	w portion of lighting data

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

im_GEsetup;

float     x = table1;   /* table1 lighting data*/
float     y = table2;   /* table2 lighting data */
float     z = table3;   /* table3 lighting data */
float     w = table4; /* table4 lighting data */

GEWAIT;
GE[GE_LIGHTDATA4].i = 0;
GE[GE_DATA].f = x;
GE[GE_DATA].f = y;
GE[GE_DATA].f = z;
GE[GE_DATA].f = w;
```

GE_LIGHTMEMPTR

This token is used to specify a lighting data address in microcode data RAM. The x parameter contains the address and the other three parameters must be zero.

Data Parameters :

data type	parameter name	parameter description
float	x	microcode data RAM address
float	y	always zero
float	z	always zero
float	w	always zero

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

im_GEsetup;

float     x = ExpTblm; /* lighting data address */
float     y = 0.0;
float     z = 0.0;
float     w = 0.0;

GEWAIT;
GE[GE_LIGHTMEMPTR].i = 0;
GE[GE_DATA].f = x;
GE[GE_DATA].f = y;
GE[GE_DATA].f = z;
GE[GE_DATA].f = w;
```

GE_LIGHTMOVEDATA

This token is used to move lighting data up or down in microcode RAM. The data parameters specify the address in microcode data RAM of the lighting data to be moved and a move up or down flag.

Data Parameters :

data type	parameter name	parameter description
int	address	address in microcode data RAM
float	moveflag	move direction flag 0.0 - move up -1.0 - move down
float	y	always zero
float	z	always zero

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

im_GEsetup;

int     address = L_W_M;   /* location in LIGHTDATA */
float   x = -1.0;          /* move down */
float   y = 0.0, z = 0.0;

GEWAIT;
GE[GE_LIGHTATTR1].i = 0;
GE[GE_DATA].i = address;
GE[GE_DATA].f = x;
GE[GE_DATA].f = y;
GE[GE_DATA].f = z;
```

GE_LIGHTPOSITION

This token is used to load lighting position data. The four data parameters contain the lighting position which will be stored at the location in microcode data RAM previously specified by the GE_LIGHTMEMPTR token.

Data Parameters :

data type	parameter name	parameter description
float	x	x portion of lighting position
float	y	y portion of lighting position
float	z	z portion of lighting position
float	w	w portion of lighting position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

im_GEsetup;

float     x = posx;     /* lighting position x */
float     y = posy;     /* lighting position y */
float     z = posz;     /* lighting position z */
float     w = 1.0;      /* 1.0 for local lighting position */

GEWAIT;
GE[GE_LIGHTDIRECTION].i = 0;
GE[GE_DATA].f = x;
GE[GE_DATA].f = y;
GE[GE_DATA].f = z;
GE[GE_DATA].f = w;
```

GE_LINESTYLE

This token is used to set the current line style parameters.

Data Parameters :

data type	parameter name	parameter description
int	style_no	line style index number
int	bitpattern	line style bitpattern

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int       style_no = 1;
int       bitpattern = 0F0F0; /* Dashed line pattern */

GEWAIT;
GE[GE_LINESTYLE].i = 0;
GE[GE_DATA].i = style_no;
GE[GE_DATA].i = bitpattern;
```

GE_LINEWIDTH

This token is used to set the current line width in pixels.

Data Parameters :

data type	parameter name	parameter description
int	width	line width value in pixels

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     width = 10;

GEWAIT;
GE[GE_LINEWIDTH].i = 0;
GE[GE_DATA].i = width;
```

GE_LMCOLOR

This token is used to set a new lighting color mode. The data parameters specify the mode. Four integer data parameters are used to specify the new color mode. The mode is specified by the values passed in the four parameters. For the LMC_COLOR mode the first parameter specifies the microcode data RAM address of the lighting mode color.

Data Parameters :

data type	parameter name	parameter description
int	lmc1	lighting mode color parameter 1
int	lmc2	lighting mode color parameter 2
int	lmc3	lighting mode color parameter 3
int	lmc4	lighting mode color parameter 4

lmc1	lmc2	lmc3	lmc4	mode
1	- 1	1	- 1	LMC_AD
- 1	- 1	1	- 1	LMC_AMBIENT
lmc addr	1	0	0	LMC_COLOR
1	- 1	- 1	- 1	LMC_DIFFUSE
- 1	- 1	- 1	1	LMC_EMISSION
- 1	- 1	- 1	- 1	LMC_NULL
- 1	1	- 1	-1	LMC_SPECULAR

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

Im_GEsetup;

 /* set for LMC_EMISSION mode */

int       lmc1 = -1, lmc2 = -1, lmc3 = -1, lmc4 = 1;

GEWAIT;
GE[GE_LMCOLOR].i = 0;
GE[GE_DATA].i = lmc1;
GE[GE_DATA].i = lmc2;
GE[GE_DATA].i = lmc3;
GE[GE_DATA].i = lmc4;
```

GE_LOADAMBIENT

This token is used to load the ambient lighting data. The data parameters specify the red, green and blue color values.

Data Parameters :

data type	parameter name	parameter description
float	r	ambient red color component
float	g	ambient green color component
float	b	ambient blue color component

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

im_GEsetup;

float     r = 0.6, g = 0.2, b = 0.7;

GEWAIT;
GE[GE_LOADAMBIENT].i = 0;
GE[GE_DATA].f = r;
GE[GE_DATA].f = g;
GE[GE_DATA].f = b;
```

GE_LOADASUM

This token is used to load the ambient sum lighting data. The data parameters specify the red, green and blue color values.

Data Parameters :

data type	parameter name	parameter description
float	r	ambient sum red color component
float	g	ambient sum green color component
float	b	ambient sum blue color component

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

im_GEsetup;

float     r = 0.6, g = 0.2, b = 0.7;

GEWAIT;
GE[GE_LOADASUM].i = 0;
GE[GE_DATA].f = r;
GE[GE_DATA].f = g;
GE[GE_DATA].f = b;
```

GE_LOADDIFFUSE

This token is used to load the diffuse reflectance lighting data. The data parameters specify the red, green and blue color values. A fourth parameter specifies the alpha value.

Data Parameters :

data type	parameter name	parameter description
float	r	diffuse red color component
float	g	diffuse green color component
float	b	diffuse blue color component
float	a	diffuse blue color component

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

im_GEsetup;

float     r = 0.6, g = 0.2, b = 0.7, a = 0.4;

GEWAIT;
GE[GE_LOADDIFFUSE].i = 0;
GE[GE_DATA].f = r;
GE[GE_DATA].f = g;
GE[GE_DATA].f = b;
GE[GE_DATA].f = a;
```

GE_LOADEMISSION

This token is used to load the material emission color lighting data. The data parameters specify the red, green and blue color values.

Data Parameters :

data type	parameter name	parameter description
float	r	emission red color component
float	g	emission green color component
float	b	emission blue color component

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

im_GEsetup;

float     r = 0.6, g = 0.2, b = 0.7;

GEWAIT;
GE[GE_LOADEMISSION].i = 0;
GE[GE_DATA].f = r;
GE[GE_DATA].f = g;
GE[GE_DATA].f = b;
```

GE_LOADGE

This token is used to write the specified data to the specified microcode data RAM address.

Data Parameters :

data type	parameter name	parameter description
int	DRAM address	Microcode data RAM address
int	DRAM data	Data to be written to data RAM

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"  /* Contains microcode data RAM address definitions */

im_GEsetup;

GEWAIT;
GE[GE_LOADGE].i = 0;
GE[GE_DATA].i = CDITHER;        /* CI dither flag address */
GE[GE_DATA].i = 1;              /* Turn on CI dithering */
```

GE_LOADLCOLOR

This token is used to load the light source color lighting data. The data parameters specify the light source and the red, green and blue color values.

Data Parameters :

data type	parameter name	parameter description
float	lsource	light source
float	r	lcolor red color component
float	g	lcolor green color component
float	b	lcolor blue color component

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

im_GEsetup;

float     lsource = 1.0, r = 0.6, g = 0.2, b = 0.7;

GEWAIT;
GE[GE_LOADLCOLOR].i = 0;
GE[GE_DATA].f = lsource;
GE[GE_DATA].f = r;
GE[GE_DATA].f = g;
GE[GE_DATA].f = b;
```

GE_LOADMATRIX

This token is used to load the specified 4 X 4 matrix data into the coordinate transformation matrix on the top of the matrix stack. The 16 data parameters are floating point values and are used to create the initial projection matrix used transform the world coordinates into normalized coordinates. The GE_MULTMATRIX token can be used to concatenate the viewing and modelling transformations to the matrix loaded by this token.

Data Parameters :

data type	parameter name	parameter description
float	a11	matrix entry a sub 11
float	a12	matrix entry a sub 12
float	a13	matrix entry a sub 13
float	a14	matrix entry a sub 14
float	a21	matrix entry a sub 21
float	a22	matrix entry a sub 22
float	a23	matrix entry a sub 23
float	a24	matrix entry a sub 24
float	a31	matrix entry a sub 31
float	a32	matrix entry a sub 32
float	a33	matrix entry a sub 33
float	a34	matrix entry a sub 34
float	a41	matrix entry a sub 41
float	a42	matrix entry a sub 42
float	a43	matrix entry a sub 43
float	a44	matrix entry a sub 44

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include  "mgr.h"
#include  "imsetup.h"
#include  "gecmds.h"
#include  "gl.h"              /* XMAXSCREEN = 1279, YMAXSCREEN = 1023 */

im_GEsetup;

/* subtract 0.5 from the min values and add 0.5 to the max values to account for the pixel
center offsetting */

float     left = -0.5, right = XMAXSCREEN + 0.5;
float     bottom = -0.5, top = YMAXSCREEN + 0.5;
float     m[4][4];
int       row, col;

for (row = 0 ; row < 4 ; row++)
    for (col = 0 ; col < 4 ; )
```

```
            m[row][col++] = 0;

    m[0][0] = 2.0/(right - left);
    m[1][1] = 2.0/(top - bottom);
    m[2][2] = -1.0;
    m[3][0] = (right + left)/(right - left);
    m[3][1] = (top + bottom)/(top - bottom);
    m[3][3] = 1.0;

    /* loads 2D orthographic projection matrix for world coordinates equal to screen coordinates
    * /

    GEWAIT;
    GE[GE_LOADMATRIX].i = 0;

    for (row = 0 ; row < 4 ; row++)
        for (col = 0 ; col < 4 ; )
                GE[GE_DATA].f = m[row][col++];
```

GE_LOADNAME

This token is used to load a name onto the top of the picking and selecting name stack. The current name at the top of the name stack is replaced and all other names are left unchanged.

Data Parameters :

data type	parameter name	parameter description
int	name	name to load onto the name stack

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     name = 4;

GEWAIT;
GE[GE_LOADNAME].i = 0;
GE[GE_DATA].i = name;
```

GE_LOADNORMAL

This token is used to load a normal matrix onto the top of the normal matrix stack. This matrix represents the inverse transpose of the upper left 3 X 3 submatrix of the 4 X 4 modeling/viewing matrix.

Data Parameters :

data type	parameter name	parameter description
float	a11	normal matrix entry a sub 11
float	a12	normal matrix entry a sub 12
float	a13	normal matrix entry a sub 13
float	a21	normal matrix entry a sub 21
float	a22	normal matrix entry a sub 22
float	a23	normal matrix entry a sub 23
float	a31	normal matrix entry a sub 31
float	a32	normal matrix entry a sub 32
float	a33	normal matrix entry a sub 33

Return Data from Microcode Data RAM :

None

Example Usage :

/* Load the upper 3 X 3 submatrix of a 4 X 4 modeling viewing matrix onto the top of the normal matrix stack. The 3 X 3 submatrix has been inverted and transposed before being sent to the Geometry Engine via the FIFO. */

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float *f;  /* f is a pointer to the 4 X 4 matrix */

GEWAIT;
GE[GE_LOADNORMAL].i = 0;
GE[GE_DATA].f = f[0];
GE[GE_DATA].f = f[1];
GE[GE_DATA].f = f[2];
GE[GE_DATA].f = f[4];
GE[GE_DATA].f = f[5];
GE[GE_DATA].f = f[6];
GE[GE_DATA].f = f[8];
GE[GE_DATA].f = f[9];
GE[GE_DATA].f = f[10];
```

GE_LOADRE

This token is used to load an RE register with a value. The RE register is specified along with the value to be placed in the register. The RE registers are defined in the include file re.h.

Data Parameters :

data type	parameter name	parameter description
int	re_reg	The RE register to be loaded
int	re_data	The data to load into the register

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "re.h"
#include "gecmds.h"

im_GEsetup;

int     re_reg = TOPSCAN;
int     re_data = 0x3FF;  /* 1 meg VRAM topscan value */

GEWAIT;
GE[GE_LOADRE].i = 0;
GE[GE_DATA].i = re_reg;
GE[GE_DATA].i = re_data;
```

GE_LOADSPECULAR

This token is used to load the specular reflectance lighting data. The data parameters specify the red, green and blue color values.

Data Parameters :

data type	parameter name	parameter description
float	r	specular red color component
float	g	specular green color component
float	b	specular blue color component

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"
#include "light.h
#include "lighting.h"

im_GEsetup;

float     r = 0.6, g = 0.2, b = 0.7;

GEWAIT;
GE[GE_LOADSPECULAR].i = 0;
GE[GE_DATA].f = r;
GE[GE_DATA].f = g;
GE[GE_DATA].f = b;
```

GE_LOADTOPMATRIX

This token is used to copy a matrix from a specified location on the special PHIGSs matrix stack to the top of the matrix stack. The PHIGSs stack occupies the bottom six matrix entries of the sixteen total matrix entries. The data parameter specifies the matrix index of the matrix which is to be copied to the top of the matrix stack. The matrix which is copied becomes the current transformation matrix. This token is not currently used by any of the graphics library calls.

Data Parameters :

data type	parameter name	parameter description
int	src_index	source matrix index

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int        src_index = 13;

GEWAIT;
GE[GE_LOADTOPMATRIX].i = 0;
GE[GE_DATA].i = src_index;
```

GE_LOADVIEWP

This token is used to set the viewport parameters which are used to transform the normalized coordinates to the device coordinates.. The host software must send ten data parameters after the token. The first three parameters are the size of the viewport and are used by the microcode to scale the normalized coordinates to get the screen coordinates. The next three parameters are the x, y and z viewport centers and are non-window relative sizes. The microcode adds the x and y origin of the window to the viewport center values and then uses the values as translation values to translate the scaled screen coordinate values. The last four data parameters are the non-window relative lower left and upper right corners of the viewport rectangle. The microcode adds the window x and y origin values to the parameters and saves the window relative viewport rectangle.

Data Parameters :

data type	parameter name	parameter description
float	vpsizex	viewport x dimension size
float	vpsizey	viewport y dimension size
float	vpsizez	viewport z dimension size
float	vpcenterx	viewport center x location
float	vpcentery	viewport center y location
float	vpcenterz	viewport center z location
int	vpleft	viewport left side location
int	vpbottom	viewport bottom location
int	vpright	viewport right side location
int	vptop	viewport top location

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "gl.h"
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     left = 0, right = XMAXSCREEN;
int     bottom = 0, top = YMAXSCREEN;
int     zmin = 0x800000, zmax = 0x7FFFFF;

/* Add 0.49 to vpsizex and vpsizey to offset rounding in the GE5 microcode */

float   vpsizex = ((float)(right-left)/2.0 + 0.49);
float   vpsizey = ((float)(top-bottom)/2.0 + 0.49);
float   vpsizez = ((float)(zmax - zmin)/2.0);
float   vpcenterx = ((float)(right+left)/2.0 + 0.5);
float   vpcentery = ((float)(top+bottom)/2.0 + 0.5);
float   vpcenterz = ((float)(zmax+zmin)/2.0 + 0.5);
```

```
GEWAIT;
GE[GE_LOADVIEWP].i = 0; /* set a full screen viewport */
GE[GE_DATA].f = vpsizex;
GE[GE_DATA].f = vpsizey;
GE[GE_DATA].f = vpsizez;
GE[GE_DATA].f = vpcenterx;
GE[GE_DATA].f = vpcentery;
GE[GE_DATA].f = vpcenterz;
GE[GE_DATA].i = vpleft;
GE[GE_DATA].i = vpbottom;
GE[GE_DATA].i = vpright;
GE[GE_DATA].i = vptop;
```

GE_LSREPEAT

This token is used to set the repeat count for the line style. The repeat count is applied to each bit of the line style bitpattern as a line is drawn.

Data Parameters :

data type	parameter name	parameter description
int	count	line style repeat count (0 - 255)

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     count = 3;

GEWAIT;
GE[GE_LSREPEAT].i = 0;
GE[GE_DATA].i = count;
```

GE_MMODE

This token is used to set the current matrix mode. The modes allowed include MSINGLE, MPROJECTION and MVIEWING. If lighting calculations are not being performed then MSINGLE mode is used. In this mode a combined modeling, viewing and projection matrix is used to transform the world coordinate data to normalized screen coordinates. If lighting calculations are being used then a modeling/viewing matrix and a separate projection matrix must be maintained. The MPROJECTION mode causes the projection matrix to be used and the MVIEWING mode causes the modeling/viewing matrix to be used.

Data Parameters :

data type	parameter name	parameter description
int	mmode	desired matrix mode 0 - MSINGLE 1 - MPROJECTION 2 - MVIEWING

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "gl.h"
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     mmode = MSINGLE;

GEWAIT;
GE[GE_MMODE].i = 0;
GE[GE_DATA].i = mmode;
```

GE_MOVE2

This token is used to update the current graphics drawing position. The 2-D position is specified in absolute world coordinates and the coordinate data parameters are specified using floating point format. The specified world coordinates are transformed to screen coordinates for use by the drawing tokens. The current graphics position is updated with the specified coordinates.

Data Parameters :

data type	parameter name	parameter description
float	mx	absolute x graphics position
float	my	absolute y graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float      mx = 4.0, my = 2.0;

GEWAIT;
GE[GE_MOVE2].i = 0;
GE[GE_DATA].f = mx;
GE[GE_DATA].f = my;
```

GE_MOVE2I

This token is used to update the current graphics drawing position. The 2-D position is specified in absolute world coordinates and the coordinate data parameters are specified using integer format.

Data Parameters :

data type	parameter name	parameter description
int	mx	absolute x graphics position
int	my	absolute y graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     mx = 4, my = 2;

GEWAIT;
GE[GE_MOVE2I].i = 0;
GE[GE_DATA].i = mx;
GE[GE_DATA].i = my;
```

GE_MOVE3

This token is used to update the current graphics drawing position. The 3-D position is specified in absolute world coordinates and the coordinate data parameters are specified using floating point format.

Data Parameters :

data type	parameter name	parameter description
float	mx	absolute x graphics position
float	my	absolute y graphics position
float	mz	absolute z graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float     mx = 4.0, my = 2.0, mz = 6.0;

GEWAIT;
GE[GE_MOVE3].i = 0;
GE[GE_DATA].f = mx;
GE[GE_DATA].f = my;
GE[GE_DATA].f = mz;
```

GE_MOVE3I

This token is used to update the current graphics drawing position. The 3-D position is specified in absolute world coordinates and the coordinate data parameters are specified using integer format.

Data Parameters :

data type	parameter name	parameter description
int	mx	absolute x graphics position
int	my	absolute y graphics position
int	mz	absolute z graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     mx = 4, my = 2, mz = 6;

GEWAIT;
GE[GE_MOVE3I].i = 0;
GE[GE_DATA].i = mx;
GE[GE_DATA].i = my;
GE[GE_DATA].i = mz;
```

GE_MOVE4

This token is used to update the current graphics drawing position. The 3-D position is specified in homogeneous world coordinates and the coordinate data parameters are specified using floating point format.

Data Parameters :

data type	parameter name	parameter description
float	mx	absolute x graphics position
float	my	absolute y graphics position
float	mz	absolute z graphics position
float	mw	absolute w graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include  "mgr.h"
#include  "imsetup.h"
#include  "gecmds.h"

im_GEsetup;

float     mx = 4.0, my = 2.0, mz = 6.0, mw = 1.0;

GEWAIT;
GE[GE_MOVE4].i = 0;
GE[GE_DATA].f = mx;
GE[GE_DATA].f = my;
GE[GE_DATA].f = mz;
GE[GE_DATA].f = mw;
```

GE_MOVE4I

This token is used to update the current graphics drawing position. The 3-D position is specified in homogeneous world coordinates and the coordinate data parameters are specified using integer format.

Data Parameters :

data type	parameter name	parameter description
int	mx	absolute x graphics position
int	my	absolute y graphics position
int	mz	absolute z graphics position
int	mw	absolute w graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     mx = 4, my = 2, mz = 6, mw = 1;

GEWAIT;
GE[GE_MOVE4I].i = 0;
GE[GE_DATA].i = mx;
GE[GE_DATA].i = my;
GE[GE_DATA].i = mz;
GE[GE_DATA].i = mw;
```

GE_MULTMATRIX

This token is used to multiply the 4 X 4 matrix on the top of the modeling/viewing/projection matrix stack with the specified 4 X 4 matrix. The 16 data parameters are floating point values and are used to concatenate additional transformation matrices with the current transformation matrix.

Data Parameters :

data type	parameter name	parameter description
float	a11	matrix entry a sub 11
float	a12	matrix entry a sub 12
float	a13	matrix entry a sub 13
float	a14	matrix entry a sub 14
float	a21	matrix entry a sub 21
float	a22	matrix entry a sub 22
float	a23	matrix entry a sub 23
float	a24	matrix entry a sub 24
float	a31	matrix entry a sub 31
float	a32	matrix entry a sub 32
float	a33	matrix entry a sub 33
float	a34	matrix entry a sub 34
float	a41	matrix entry a sub 41
float	a42	matrix entry a sub 42
float	a43	matrix entry a sub 43
float	a44	matrix entry a sub 44

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float     tx = 10.0, ty = 20.0, tz = 5.0;
float     m[4][4];
int       row, col;

for (row = 0 ; row < 4 ; row++)
    for (col = 0 ; col < 4 ; )
        m[row][col++] = 0;

/* Multiply current matrix by translation matrix */

m[0][0] = 1.0;
m[1][1] = 1.0;
m[2][2] = 1.0;
```

```
m[3][0] = tx;
m[3][1] = ty;
m[3][2] = tz;
m[3][3] = 1.0;

GEWAIT;
GE[GE_MULTMATRIX].i = 0;

for (row = 0 ; row < 4 ; row++)
    for (col = 0 ; col < 4 ; )
            GE[GE_DATA].f = m[row][col++];
```

GE_MULTNORMAL

This token is used to multiply the top of the normal matrix stack by the normal matrix sent as data. This matrix represents the inverse transpose of the upper left 3 X 3 submatrix of the 4 X 4 modeling/viewing matrix.

Data Parameters :

data type	parameter name	parameter description
float	a11	normal matrix entry a sub 11
float	a12	normal matrix entry a sub 12
float	a13	normal matrix entry a sub 13
float	a21	normal matrix entry a sub 21
float	a22	normal matrix entry a sub 22
float	a23	normal matrix entry a sub 23
float	a31	normal matrix entry a sub 31
float	a32	normal matrix entry a sub 32
float	a33	normal matrix entry a sub 33

Return Data from Microcode Data RAM :

None

Example Usage :

/* Multiply the top of the normal matrix stack by the upper 3 X 3 submatrix of a 4 X 4 modeling viewing matrix. The 3 X 3 submatrix has been inverted and transposed before being sent to the Geometry Engine via the FIFO. */

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float *f;  /* f is a pointer to the 4 X 4 matrix */

GEWAIT;
GE[GE_MULTNORMAL].i = 0;
GE[GE_DATA].f = f[0];
GE[GE_DATA].f = f[1];
GE[GE_DATA].f = f[2];
GE[GE_DATA].f = f[4];
GE[GE_DATA].f = f[5];
GE[GE_DATA].f = f[6];
GE[GE_DATA].f = f[8];
GE[GE_DATA].f = f[9];
GE[GE_DATA].f = f[10];
```

GE_NORMAL

This token is used to update the current normal vector used in lighting calculations. The data parameters consist of three world coordinate floating point values.

Data Parameters :

data type	parameter name	parameter description
float	vx	normal vector x value
float	vy	normal vector y value
float	vz	normal vector z value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float       vx = 10.0, vy =15.0, vz = 7.0;

GEWAIT;
GE[GE_NORMAL].i = 0;
GE[GE_DATA].f = vx;
GE[GE_DATA].f = vy;
GE[GE_DATA].f = vz;
```

GE_PASSTHROUGH

This token is used to pass an integer through the Geometry Pipeline for use in parsing feedback data. This token should only be used in feedback mode.

Data Parameters :

data type	parameter name	parameter description
int	passtoken	a 16 bit integer passthrough token

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     passtoken = 0xABCD; /* feedback marker */

GEWAIT;
GE[GE_PASSTHROUGH].i = 0;
GE[GE_DATA].i = passtoken;
```

GE_PATTERN

This token is used to disable pattern usage which had been enabled by a previous GE_SETPATTERN token. Patterns are used during textured polygon drawing.

Data Parameters :

data type	parameter name	parameter description
int	pat_flag	pattern disable flag

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     pat_flag = 0;

GEWAIT;
GE[GE_PATTERN].i = 0;
GE[GE_DATA].i = pat_flag;
```

GE_PICKMODE

This token is used to place the GE into picking mode. The pick mode type is set to 7 for pick all, after this token is received by the microcode.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_PICKMODE].i = 0;
```

GE_PICKTYPE

This token is used to change the picking mode type. This token would be sent after the GE_PICKMODE token had been sent. This token allows the host software to control the amount of pick data that is being saved and therefore must be processed when the Geometry Engine generates a graphics interrupt to notify the host that the pick buffer is full and needs to be read. Four different pick mode types are supported. Pick none causes all of the pick data to be discarded. Pick first causes all pick data after the first pick event to be discarded. Pick last causes the current pick data to overwrite the pick data in the pick buffer preserving only the last pick data. For pick first and pick last the GE_ENDPICKMODE token is sent to read the pick data buffer. Pick all causes all pick data to be saved in the buffer. When the buffer is full the microcode generates a graphics interrupt to notify the host. The GE_ENDPICKMODE token is sent to read the pick data placed in the pick buffer since the last interrupt was processed.

For pick last if the namestack in the microcode is greater than the microcode buffer the namestack is sent to the host. The host software must be prepared to allow this to be done.

Data Parameters :

data type	parameter name	parameter description
int	picktype	specifies the pick mode data that will be saved 1 - pick none 3 - pick first 5 - pick last 7 - pick all

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     picktype = 3; /* pick first */

GEWAIT;
GE[GE_PICKTYPE].i = 0;
GE[GE_DATA].i = picktype;
```

GE_PIXTYPE

This token is used to set the Raster Engine PIXTYPE register which determines the pixel formats for pixel data written to the Frame Buffer bitplanes. The pixel types include 4, 8 or 12 bit Color Index (CI) pixels and 8, 12 or 24 bit RGB pixels. Refer to the Raster Subsystem chapter for additional details on the pixel type formatting that is done by the RE2. For the Color Index pixels the GE_COLOR or GE_COLORF tokens are used to specify the color index color value. This value will be used by the Display Subsystem to index into a color map to get a 24 bit RGB color value for display. For the RGB pixels the red, green and blue color components are specified with the GE_RGBCOLOR token. These colors are used directly for display. To allow drawing to take place into the Frame Buffer bitplanes the GE_DRAWMODE token must be used to select a drawing mode of NORMALDRAW.

Pixel type 0 selects either 24 bit or 8 bit RGB pixels. For the base configuration with 8 Frame Buffer bitplanes the ENABRGB register must be set to one for 8 bit RGB pixels. If the ENABRGB register is set to zero then the RE2 will attempt to format the pixels format as 24 bit RGB but only the Red color component will be written. For the enhanced adapter pixel type 0 will always be 24 bit RGB regardless of the ENABRGB register setting.

Pixel type 1 selects 12 bit RGB double buffered pixels. This pixel type is invalid on the base adapter. The most significant nibble of each color component is duplicated into the lower nibble and the PIXMASK determines which buffer is written.

Pixel type 2 selects 8 bit Color Index single buffered pixels on the base adapter and it selects 12 bit Color Index double buffered pixels on the enhanced adapter. The 12 bit Color Index color will be duplicated into the upper 12 bits and all 24 bits will be written. The PIXMASK determines which bits are written.

Pixel type 3 selects 4 bit Color Index double buffered pixels on both adapter configurations. The 4 bit Color Index color will be duplicated into the upper 4 bits of the right most byte and all 8 bits will be written. The PIXMASK determines which bits are written.

The PIXMASK set with the GE_PIXWRITEMASK token determines which bits in the Frame Buffer bitplanes are written to. The following values can be used for the PIXWRITEMASK:

0x0000FF	for writes to all the base adapter Frame Buffer bitplanes
0xFFFFFF	for writes to all the enhanced adapter Frame Buffer bitplanes
0xFFFFFF	for writes to all bits of PIXTYPE = 0 (24 bit RGB)
0x0000FF	for writes to all bits of PIXTYPE = 0 (8 bit RGB)
0x0F0F0F	for writes to the front buffer with PIXTYPE = 1 (12 bit RGB)
0xF0F0F0	for writes to the back buffer with PIXTYPE = 1 (12 bit RGB)
0x000FFF	for writes to the front buffer with PIXTYPE = 2 (8 or 12 bit CI)
0xFFF000	for writes to the back buffer with PIXTYPE = 2 (8 or 12 bit CI)
0x00000F	for writes to the front buffer with PIXTYPE = 3 (4 bit CI)
0x0000F0	for writes to the back buffer with PIXTYPE = 3 (4 bit CI)

Data Parameters :

data type	parameter name	parameter description
int	pixtype	specified 2 bit pixel type
	bits1:0	meaning

00	24 bit RGB pixel or 8 bit RGB, single buffered
01	12 bit RGB pixel, double buffered
10	8 bit CI pixel, single buffered, base adapter 12 bit CI pixel, double buffered, enhanced adapter
11	4 bit CI pixel, double buffered

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     pixtype = 3;  /* 4 bit Color Index pixel type */

GEWAIT;
GE[GE_PIXTYPE].i = 0;
GE[GE_DATA].i = pixtype;
```

GE_PIXWRITEMASK

This token is used to set the Raster Engine PIXMASK register. This register is 24 bits wide and is a write mask for the frame buffer bitplanes. Each bitplane has a corresponding mask bit in the register. Mask bits set to 1 enable writes to the corresponding bitplane. Mask bits set to 0 disable writes to the corresponding bitplane. Refer to the GE_PIXTYPE token for the various Frame Buffer pixel types and suitable values to set the PIXMASK to.

Data Parameters :

data type	parameter name	parameter description
int	pixmask	Frame buffer pixel write mask

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "lmsetup.h"
#include "gecmds.h"

im_GEsetup;

/* set write mask for 12 bit CI front buffer writes */

int     pixmask = 0x000FFF;

GEWAIT;
GE[GE_PIXWRITEMASK].i = 0;
GE[GE_DATA].i = pixmask;
```

GE_PNT2

This token is used to draw a point at the specified 2-D position. The position is specified in absolute world coordinates and the coordinate data parameters are specified using floating point format. After the point is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
float	px	absolute x graphics position
float	py	absolute y graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float     px = 4.0, py = 2.0;

GEWAIT;
GE[GE_PNT2].i = 0;
GE[GE_DATA].f = px;
GE[GE_DATA].f = py;
```

GE_PNT2I

This token is used to draw a point at the specified 2-D position. The position is specified in absolute world coordinates and the coordinate data parameters are specified using integer format. After the point is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
int	px	absolute x graphics position
int	py	absolute y graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     px = 4, py = 2;

GEWAIT;
GE[GE_PNT2I].i = 0;
GE[GE_DATA].i = px;
GE[GE_DATA].i = py;
```

GE_PNT3

This token is used to draw a point at the specified 3-D position. The position is specified in absolute world coordinates and the coordinate data parameters are specified using floating point format. After the point is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
float	px	absolute x graphics position
float	py	absolute y graphics position
float	pz	absolute z graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include  "mgr.h"
#include  "imsetup.h"
#include  "gecmds.h"

im_GEsetup;

float     px = 4.0, py = 2.0, pz = 6.0;

GEWAIT;
GE[GE_PNT3].i = 0;
GE[GE_DATA].f = px;
GE[GE_DATA].f = py;
GE[GE_DATA].f = pz;
```

GE_PNT3I

This token is used to draw a point at the specified 3-D position. The position is specified in absolute world coordinates and the coordinate data parameters are specified using integer format. After the point is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
int	px	absolute x graphics position
int	py	absolute y graphics position
int	pz	absolute z graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     px = 4, py = 2, pz = 6;

GEWAIT;
GE[GE_PNT3I].i = 0;
GE[GE_DATA].i = px;
GE[GE_DATA].i = py;
GE[GE_DATA].i = pz;
```

GE_PNT4

This token is used to draw a point at the specified 3-D position. The position is specified in absolute homogeneous coordinates and the coordinate data parameters are specified using floating point format. After the point is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
float	px	absolute x graphics position
float	py	absolute y graphics position
float	pz	absolute z graphics position
float	pw	absolute w graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float     px = 4.0, py = 2.0, pz = 6.0, pw = 1.0;

GEWAIT;
GE[GE_PNT4].i = 0;
GE[GE_DATA].f = px;
GE[GE_DATA].f = py;
GE[GE_DATA].f = pz;
GE[GE_DATA].f = pw;
```

GE_PNT4I

This token is used to draw a point at the specified 3-D position. The position is specified in absolute homogeneous coordinates and the coordinate data parameters are specified using integer format. After the point is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
int	px	absolute x graphics position
int	py	absolute y graphics position
int	pz	absolute z graphics position
int	pw	absolute w graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     px = 4, py = 2, pz = 6, pw = 1;

GEWAIT;
GE[GE_PNT4I].i = 0;
GE[GE_DATA].i = px;
GE[GE_DATA].i = py;
GE[GE_DATA].i = pz;
GE[GE_DATA].i = pw;
```

GE_POLYGON

This token is used to enter polygon mode. All vertex data after this token is treated as polygon vertices until a GE_ENDPOLYGON or GE_ENDOLDPOLYGON token is received. These tokens end polygon mode.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_POLYGON].i = 0;
```

GE_POPMATRIX

This token instructs the microcode to pop the top matrix off of the matrix stack. The popped matrix is discarded and all other matrices on the stack move up one level in the stack.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_POPMATRIX].i = 0;
```

GE_POPNAME

This token is used to pop the top name off the picking and selecting name stack. All other names on the stack move up one level.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_POPNAME].i = 0;
```

GE_PUSHMATRIX

This token instructs the microcode to push the matrices on the matrix stack down one level. After the push, the top of the stack contains a copy of the pushed top matrix.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_PUSHMATRIX].i = 0;
```

GE_PUSHNAME

This token is used to push a name onto the top of the picking and selecting name stack. The current names on the name stack are pushed down and the specified name is placed on the top of the stack.

Data Parameters :

data type	parameter name	parameter description
Int	name	name to load onto the name stack

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int         name = 2;

GEWAIT;
GE[GE_PUSHNAME].i = 0;
GE[GE_DATA].i = name;
```

GE_PUSHV

This token is used to copy the NURBS coefficients calculated with the GE_SETV token to a different location in the microcode data RAM.

Data Parameters :

data type	parameter name	parameter description
float	dummy	dummy parameter always zero

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float           dummy = 0.0;

GEWAIT;
GE[GE_PUSHV].i = 0;
GE[GE_DATA].f = dummy;
```

GE_RASTEROP

This token is used to set the raster operation mode which is used to determine the bitwise logical operation which is used to combine the pixels being written (src) and the pixels already in the selected bitplanes (dest). Logical operations can be performed on the frame buffer bitplanes, the auxiliary bitplanes and the Window ID bitplanes as selected by the GE_RWMODE token. The default raster operation mode is 3 (Copy). In the description of the modes the term COMP is used to indicate a 1's complement operation. The AND, OR and XOR terms are the standard logical bitwise operators.

For the GE_WRITEPIXELS, the GE_RECTWRITE, the GE_WRITEBLOCK and the GE_RECTCOPY tokens if pixel packing or pixel x zooming is used then the raster op mode must be set to 3 (SRC_COPY).

Data Parameters :

data type	parameter name	parameter description
int	ropmode	desired raster operation mode
int	half_closed_flag	determines if half closed line is drawn -1 - for all modes except 6 and 9 1 - for modes 6 and 9

ropmode (hex)	name	value written to destination
0	Clear	zero
1	AND	src AND dest
2	AND Reverse	src AND (COMP dest)
3	Copy	src
4	AND Inverted	(COMP src) AND dest
5	No op	dest
6	XOR	src XOR dest
7	OR	src OR dest
8	NOR	(COMP src) AND (COMP dest)
9	Equivalent	(COMP src) XOR dest
A	Invert	COMP dest
B	OR Reverse	src OR (COMP dest)
C	Copy Inverted	COMP src
D	OR Inverted	(COMP src) OR dest
E	NAND	(COMP src) OR (COMP dest)
F	Set	One

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "gl.h"                /* LO_XOR = 6 and LO_XNOR = 9 */
```

```
im_GEsetup;

int         ropmode = LO_XOR;   /* LO_XOR mode = 6 */

GEWAIT;
GE[GE_RASTEROP].i = 0;
GE[GE_DATA].i = ropmode;
if (ropmode == LO_XOR || ropmode == LO_XNOR)
    GE[GE_DATA].i = 1;
else
    GE[GE_DATA].i = -1;
```

GE_RDRAW2

This token is used to draw a line from the current graphics drawing position to the specified 2-D position. The destination position is specified in relative world coordinates and the coordinate data parameters are specified using floating point format. After the line is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
float	mx	relative x graphics position
float	my	relative y graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float     mx = 4.0, my = 2.0;

GEWAIT;
GE[GE_RDRAW2].i = 0;
GE[GE_DATA].f = mx;
GE[GE_DATA].f = my;
```

GE_RDRAW2I

This token is used to draw a line from the current graphics drawing position to the specified 2-D position. The destination position is specified in relative world coordinates and the coordinate data parameters are specified using integer format. After the line is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
int	mx	relative x graphics position
int	my	relative y graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     mx = 4, my = 2;

GEWAIT;
GE[GE_RDRAW2I].i = 0;
GE[GE_DATA].i = mx;
GE[GE_DATA].i = my;
```

GE_RDRAW3

This token is used to draw a line from the current graphics drawing position to the specified 3-D position. The destination position is specified in relative world coordinates and the coordinate data parameters are specified using floating point format. After the line is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
float	mx	relative x graphics position
float	my	relative y graphics position
float	mz	relative z graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float     mx = 4.0, my = 2.0, mz = 6.0;

GEWAIT;
GE[GE_RDRAW3].i = 0;
GE[GE_DATA].f = mx;
GE[GE_DATA].f = my;
GE[GE_DATA].f = mz;
```

GE_RDRAW3I

This token is used to draw a line from the current graphics drawing position to the specified 3-D position. The destination position is specified in relative world coordinates and the coordinate data parameters are specified using integer format. After the line is drawn the current graphics position is updated to the destination position.

Data Parameters :

data type	parameter name	parameter description
int	mx	relative x graphics position
int	my	relative y graphics position
int	mz	relative z graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include  "mgr.h"
#include  "imsetup.h"
#include  "gecmds.h"

im_GEsetup;

int       mx = 4, my = 2, mz = 6;

GEWAIT;
GE[GE_RDRAW3I].i = 0;
GE[GE_DATA].i = mx;
GE[GE_DATA].i = my;
GE[GE_DATA].i = mz;
```

GE_READBLOCK

This token is used to read a rectangular block of pixels from the selected bitplanes to the host using Raster Engine to Host DMA transfers. The microcode allows the host to read the rectangle as one large block or as multiple smaller blocks. For the multiple blocks method the host can divide the rectangle along the y axis. The number of rows in each block can be different but the number of pixels in the x direction must be the same in each block. Each block can also be read into different host buffers. As the DMA transfers are performed the rectangle is read beginning at the lower left corner specified by the x, y parameters and progresses to the right and up the screen. The GE_READBLOCK should set the READ bit in the flags word sent to the do_pixel_dma function described in the Pixel DMA Support paragraphs of the Geometry Subsystem chapter. This will cause a read block mode DMA transfer to be done.

The GE_READSOURCE token determines the source bitplanes which are to be read. The normal use of the GE_READBLOCK token is to read from the frame buffer bitplanes, however any of the bitplanes which can be selected with the GE_READSOURCE token can be read from. If the frame buffer is selected as the source for the read the GE_READBUF token specifies whether the front or back buffer is read and the GE_PIXTYPE token tells the RE2 what type of pixel data is being read.

The upacmode parameter is ignored by the microcode since the RE2 does not pack the pixel data for pixel read operations. The host buffer should be declared as an array of type long to receive the pixel data read from the selected bitplanes. Regardless of the pixel format each long word in the host buffer will receive a single pixel value which is right justified. Refer to the RE2 DMA support paragraphs in the Raster Subsystem chapter for a description of the pixel formats received from the various source bitplanes.

Data Parameters :

Block Mode

data type	parameter name	parameter description
int	upacmode	pixel packing mode for RE2 0 - 1 pixel/word
int	x	lower left x absolute screen coordinate of rect
int	xlen	x length of rectangle
int	y	lower left y absolute screen coordinate of rect
int	dma_flag	flag indicating if a GE interrupt is used to indicate the block dma transfer has completed. 1 - GE int used to signal block dma done -1 - GE int not used to signal dma done
int	ylen	y length of rectangle (this parameter may be repeated if the rectangle is in multiple blocks)
int	done_flag	flag which indicates the operation is done. -1 indicates dma done

Parameter Checks Performed by Microcode for Block Mode

x < -1280	Not checked and will produce undefined results
x < 0	Draw only the pixels which are located past 0 in the rectangle
x > 1280	No error condition reported and no pixels drawn into bitplanes

y < 0	Not checked and will produce undefined results
ylen < 1	First ylen must be >= 1 or finish 1 flag never gets set

Return Data from Microcode Data RAM :

address	description
DMA_FLAG	indicates if the microcode is ready to continue with the DMA transfers - 1 cancel the token execution and do no DMA operations 0 continue with token execution and do the DMA operations
HQMSAV	contains the last HQ MAR reg value saved with the GE_HQMSAV token

Example Usage :

```
#include  "mgr.h"
#include  "gecmds.h"
#include  "ge5_glob.h"

long      token = GE_READBLOCK;
int       x = 200, y = 400, xlen = 500, ylen = 20;
long      pixsize = 4;        /* each pixel occupies 4 bytes in the host buffer */
long      upacmode = 0;       /* 1 pixels/long word */
short     pixbuf[10000];
int       flags = READ;

/* This example shows the reading of a rectangle using the do_pixel_dma function described in
the Pixel DMA Support paragraphs in the Programming Considerations section of the Geometry
Subsystem chapter. */

/* flags would be just READ so do_pixel_dma routine would do a block mode read transfer */

if (do_pixel_dma (token, pixbuf, pixsize, flags, x, xlen, y, ylen) == ERROR)
    printf ("GE_READBLOCK error!\n");
```

GE_READBUF

This token is used to select the front or back buffer when doing pixel reads or rectangle reads from the frame buffer bitplanes. This token only affects the GE_READPIXELS, the GE_READPIXDMA, the GE_RECTREAD, the READBLOCK and the GE_RECTCOPY tokens when the GE_READSOURCE token specifies the read source bitplanes as the Frame Buffer bitplanes (source = 0).

Data Parameters :

data type	parameter name	parameter description
int	buffer	front or back buffer to be read from 0 - front buffer 1 - back buffer

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     buffer = 1;         /* back buffer */

GEWAIT;
GE[GE_READBUF].i = 0;
GE[GE_DATA].i = buffer;
```

GE_READPIXDMA

This token is used to read along a single scan line a specified number of pixels, up to a maximum of 1280, from the bitplanes selected by the GE_READSOURCE token. The normal usage of this token is to read Color Index or RGB pixels from the Frame Buffer bitplanes. After the GE_READPIXDMA token is received by the microcode it gets the current character position to use as the starting x and y screen location for the pixel read. The microcode performs an RE2 to Host Buffer DMA transfer from the selected bitplanes. It then writes the number of pixels it has read into the NUMPIX location in microcode data RAM and sets the finish 1 flag.

The host should set the READ and SINGLE bits in the flags parameter and then call the do_pixel_dma function described in the Pixel DMA Support section earlier in the Geometry Subsystem chapter. This causes a single mode read DMA transfer to be performed. Once the function call has completed the host should read the NUMPIX location in the microcode data RAM to determine the actual number of pixels read. To read the NUMPIX location the host clears finish flag 0, sends the GE_FINISH0 token and then polls finish flag 0 until it is set by the microcode. Once finish flag 0 has been set it is safe to access the GE5 data RAM. The host sets the HQ MAR register and does a DRAM_RD to get the value stored in the NUMPIX location. The value could be less than the requested pix_count if the read would have gone past the right edge of the screen. The pixels read outside of the current viewport up to the right screen edge are returned but are undefined. This means they may not match the pixel format of the pixels read inside the viewport.

Data Parameters :

data type	parameter name	parameter description
int	pix_count	number of pixels to be read

Data Parameter Checks Performed by the Microcode

pix_count < 1	The NUMPIX value is set to 0 and no pixels are read
cpos x < 0	Only pixels >= 0 will be displayed
cpos invalid	If cpos is invalid then the NUMPIX value is set to 0 and no pixels are read and the DMA_FLAG is set to DMA_CANCEL cpos is invalid if cpos x > right edge of the screen or if cpos x > viewport right edge on previous token execution

Return Data from Microcode Data RAM :

address	description
NUMPIX	Contains the actual number of pixels transfered to the host buffer

Example Usage :

```
#include "mgr.h"
#include "gecmds.h"
#include "ge5_glob.h"

long    token = GE_READPIXDMA;
int     xlen = 500, ylen = 1;
long    pixsize = 4;        /* each pixel occupies 4 bytes in the host buffer */
```

```
long        upacmode = 0;        /* 1 pixels/long word */
short       pixbuf[500];
int         flags = READ | SINGLE;

/* This example shows the reading of a rectangle using the do_pixel_dma function described in
the Pixel DMA Support paragraphs in the Programming Considerations section of the Geometry
Subsystem chapter. */

/* flags would be READ and SINGLE so do_pixel_dma routine would do a single mode read
transfer */

if (do_pixel_dma (token, pixbuf, pixsize, flags, x, xlen, y, ylen) == ERROR)
    printf ("GE_READPIXDMA error!\n");

FINISH_WR(FINISH0_OFF, 0);  /* clear finish flag 0 */

GE[GE_FINISH0].i = 0;  /* Tell microcode to set the Finish 0 flag */

while (1) {          /* wait for microcode to set finish flag 0 */
    if (FINISH_POLL(FINISH0_OFF)
        break;
    for (i = 0; i < 10 ; ++i) ;    /* Don't check it every pass */
}

GL_HQM_WR(NUMPIX);      /* set HQ MAR register and HQMSAV to NUMPIX */
DRAM_RD(NUMPIX, pix_count);
```

GE_READPIXELS

This token is used to read along a single scan line a specified number of pixels, up to a maximum of 1280, from the bitplanes selected by the GE_READSOURCE token. The normal usage of this token is to read Color Index or RGB pixels from the Frame Buffer bitplanes. After the GE_READPIXELS token is received by the microcode it gets the current character position to use as the starting x and y screen location for the pixel read. The microcode performs an RE2 to GE5 Data RAM DMA transfer from the selected bitplanes. It then writes the number of pixels it has read into the NUMPIX location in microcode data RAM.

The host must wait for the microcode to finish the GE_READPIXELS token before it can read the GE5 Data RAM to get the pixel values. The host uses finish flag 0 to synchronize it's access of the GE5 Data RAM. It clears finish flag 0 and then sends down the GE_FINISH0 token. The host then polls finish flag 0 until it is set by the microcode. After finish flag 0 has been set the host then reads NUMPIX to get the pixel count and reads that number of pixels from the GE5 Data RAM pixel buffer which begins at PIXELMEM. Once the pixel buffer has been read the host clears finish flag 0 and sets the GEPIXELS flag to -1 indicating that the pixel buffer is empty.

Data Parameters :

data type	parameter name	parameter description
int	pix_count	the number of pixels to read

Data Parameter Checks Performed by the Microcode

pix_count < 1	The NUMPIX value is set to 0 and no pixels are read
cpos x < 0	Only pixels >= 0 will be displayed
cpos invalid	If cpos is invalid then the NUMPIX value is set to 0 and no pixels are read cpos is invalid if cpos x > right edge of the screen or if cpos x > viewport right edge on previous token execution

Return Data from Microcode Data RAM :

address	description
GEPIXELS	Flag in microcode data ram which is set by ucode to indicate that pixels remain in the pixel buffer. If a context switch occurs and the flag is still set, the pixel buffer must be saved during the context switch. The host clears this location after all the pixels have been read.
NUMPIX	Contains the actual number of pixels to be read from the DRAM buffer
PIXELMEM	The starting address of the pixel data buffer in microcode data RAM

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"

im_GEsetup;
```

```
int         pix_count = 250, i;
long        pix_buffer[250];
int         src, *dest;

GE[GE_READPIXELS].i = 0;
GE[GE_DATA].i = pix_count;

FINISH_WR(FINISH0_OFF, 0);  /* clear finish flag 0 */

GE[GE_FINISH0].i = 0;  /* Tell microcode to set the Finish 0 flag */

while (1) {         /* wait for microcode to set finish flag 0 */
    if (FINISH_POLL(FINISH0_OFF)
        break;
    for (i = 0; i < 10 ; ++i) ;    /* Don't check it every pass */
}

GL_HQM_WR(NUMPIX);      /* set HQ MAR register and HQMSAV to NUMPIX */
DRAM_RD(NUMPIX, pix_count);

GL_HQM_WR(PIXELMEM);  /* set HQ MAR register and HQMSAV to PIXELMEM */

src = PIXELMEM;
dest = pix_buffer;

for (i = 0 ; i < pix_count < i++) {
    DRAM_RD(src++, *dest++);
    if (src % HQM_PG_SIZE) == 0)  /* next page in ucode DRAM ? */
        GL_HQM_WR(src);                /* yes, set new page in HQ MAR */
}

/* Clear the finish flag 0 so context switch knows were not trying to still read ucode DRAM */

FINISH_WR(FINISH0_OFF, 0);

/* Turn off GEPIXELS flag so pixel area is not saved by next context switch. */

GE[GE_LOADGE].i = 0;
GE[GE_DATA].i = GEPIXELS;
GE[GE_DATA].i = -1;
```

GE_READRGB

This token is no longer used. The GE_READPIXELS token does the necessary read operation. The RGB data must be unpacked from the pixel data which is read by GE_READPIXELS. The RGB data is in the pixel long words in BGR order with the Red byte being the right most byte in the word.

GE_READSOURCE

This token is used to set the source bitplanes for pixel reads. The bitplanes which can be read include the frame buffer bitplanes, the pup bitplanes, the uaux bitplanes, the window ID bitplanes and the Z buffer bitplanes as well as combinations of these as shown below. The GE_READSOURCE token selects the source bitplanes for the GE_READPIXELS, the GE_READBLOCK, the GE_READPIXDMA, the GE_RECTREAD and the GE_RECTCOPY tokens.

Data Parameters :

data type	parameter name	parameter description
int	source	desired read source 0 - Frame buffer bitplanes 1 - PUP bitplanes 2 - UAUX bitplanes 3 - Z buffer bitplanes 4 - Window ID (WID) bitplanes 5 - invalid 6 - Frame buffer port 7 - Z buffer port

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     source = 0;   /* source bitplanes are the frame buffer bitplanes */

GEWAIT;
GE[GE_READSOURCE].i = 0;
GE[GE_DATA].i = source;
```

GE_RECTCOPY

This token is used to copy a rectangular block of pixels from the source bitplanes specified by GE_READSOURCE to the destination bitplanes specified by GE_RWMODE. The microcode uses RE2 to GE5 data RAM DMA to read a line of the source rectangle and then does a GE5 data RAM to RE2 DMA transfer to write the line to the destination rectangle. This sequence is repeated until all the lines have been transfered. The source rectangle must be on the physical screen but not necessarily constrained to the current window. If the source rectangle is from the Frame Buffer bitplanes the GE_READBUF token determines which buffer is read.

The microcode expects the GE_RECTCOPY token to be followed by the lower left y location of the source rectangle followed by the lower left y location for the destination rectangle. The y direction length in pixels is then specified. These three tokens are followed by the source rectangle lower left x location, the destination rectangle x lower left location and the x length value. The destination rectangle pixel writes are affected by the screen mask clipping and by the WID checking and pattern masking if enabled. Refer to the RE2 DMA support section of the Raster Subsystem chapter for additional details. The logical operation specified by the GE_RASTEROP token can also be used to affect the destination rectangle writes. The destination rectangle can also be a zoomed copy of the source rectangle. The GE_ZOOMFACTOR token is used to specify the x and y zoom factors. If x zooming is used then the raster operation must be set to 3 (SRC_COPY).

Data Parameters :

data type	parameter name	parameter description
int	y1	lower left y location of src rectangle
int	newy	lower left y location of dest rectangle
int	ylen	y direction pixel length of src rectangle
int	x1	lower left x location of src rectangle
int	newx	lower left x location of dest rectangle
int	xlen	x direction pixel length of src rectangle

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     x1 = 200, y1 = 300, xlen = 100, ylen = 200;
int     newx = 400, newy = 600;

GEWAIT;
GE[GE_RECTCOPY].i = 0;
GE[GE_DATA].i = y1;
GE[GE_DATA].i = newy;
GE[GE_DATA].i = ylen;
```

```
GE[GE_DATA].i = x1;
GE[GE_DATA].i = newx;
GE[GE_DATA].i = xlen;
```

GE_RECTREAD

This token is used to read pixel data from a small rectangular area where the number of pixels in the rectangle is less than or equal to 1280. After receiving the token and data parameters the microcode does the necessary DMA transfers from the Raster Engine bitplanes to the microcode data RAM. The bitplanes to be read are specified by the GE_READSOURCE and the GE_READBUF tokens. If the pixel data which is being read is not right justified the Raster Engine will shift the pixel data to the right so that the LSB bit of the pixel is always in bit 0. This problem is encountered when reading from the back buffer of the frame buffer bitplanes. After the RE to DRAM DMA transfers are completed the microcode stores the pixel count in the NUMPIX data RAM location. The starting address of the microcode data RAM pixel buffer is PIXELMEM. The microcode also sets the GEPIXELS flag indicating that the pixel buffer contains data which has not been read by the host.

Before the host can read the pixels from the microcode data RAM to the host buffer the host must be sure that the microcode has completed the pixel DMA transfers described above. This synchronization is accomplished by sending the GE_FINISH0 token down the FIFO to cause the microcode to set finish flag 0 and to stall when it is ready for the host to read the microcode data RAM. After finish flag 0 is set the host reads NUMPIX to get the pixel count and then reads that number of pixels from the microcode data RAM beginning at address PIXELMEM. After all of the pixels have been read the host clears the GEPIXELS flag so that the pixel buffer will not be saved on a context switch unnecessarily.

Data Parameters :

data type	parameter name	parameter description
int	x1	lower left x absolute screen coordinate of rectangle
int	xlen	x direction pixel length of rectangle
int	y1	lower left y absolute screen coordinate of rectangle
int	ylen	y direction pixel length of rectangle

Return Data from Microcode Data RAM :

address	description
GEPIXELS	indicates if the pixel data area needs to be saved during a context switch
NUMPIX	the number of pixels to be read from microcode data RAM pixel buffer
PIXELMEM	pixel buffer start address in microcode data RAM

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"

im_GEsetup;

int     x1 = 200, y1 =400;
int     xlen = 50, ylen = 5;
int     pix_count, i;
int     pix_buffer[250];
```

```
int         src, *dest;

GEWAIT;
GE[GE_RECTREAD].i = 0;
GE[GE_DATA].i = x1;
GE[GE_DATA].i = xlen;
GE[GE_DATA].i = y1;
GE[GE_DATA].i = ylen;

FINISH_WR(FINISH0_OFF, 0);  /* make sure finish flag 0 is cleared */

/* Tell microcode to set finish flag 0 which will indicate that RECTREAD has completed */

GE[GE_FINISH0].i = 0;

while (1) {         /* wait for microcode to set finish flag 0 */
    if (FINISH_POLL(FINISH0_OFF))
        break;
    for (i = 0 ; i < 10 ; i++) ;   /* Let's delay a little while before the next poll */
}

GL_HQM_WR(NUMPIX);          /* set HQ MAR register to NUMPIX address */
DRAM_RD(NUMPIX, pix_count);         /* read the pixel count */

GL_HQM_WR(PIXELMEM);        /* set HQ MAR register to start address of pixel buffer */

src = PIXELMEM;
dest = pix_buffer;

for (i = 0 ; i < pix_count ; i++) {
    DRAM_RD(src++, *dest++);
    if (src % HQM_PG_SIZE) == 0)  /* next ucode data ram page ? */
        GL_HQM_WR(src);                 /* yes, reset HQ MAR */
}

FINISH_WR(FINISH0_OFF, 0);  /* make sure finish flag 0 is cleared when were done */

/* Turn off GEPIXELS flag so pixel area is not saved by next context switch. */

GE[GE_LOADGE].i = 0;
GE[GE_DATA].i = GEPIXELS;
GE[GE_DATA].i = -1;
```

GE_RECTWRITE

This token is used to write a small rectangular block of pixels thru the FIFO without using host to frame buffer DMA. The rectangle can specify multiple scan lines but the total number of pixels in the rectangle cannot exceed 1280. This limit is imposed by the size of the pixel buffer in the GE5 data RAM. This token supports the use of packed pixels but does not support pixel zooming in either the x or y direction. The first parameter specifies the dma word count a line if it started on a long word boundary. The second parameter specifies the total number of pixel data words to be sent down the FIFO and cannot exceed 1280. The third parameter is the pixel packing mode and represents the number of pixels/long -1. The fourth parameter is the starting x location. The fifth parameter is the number of pixels to be written in the x direction on each scan line. The sixth parameter is the starting y location. The seventh parameter is the number of rows in the rectangle. The eight parameter is a margin flag which is used by the microcode to calculate how many long words are used for each line. The final four parameters represent the starting pixel offset for each group of four lines in the rectangle.

The host sends the token and the data parameters down the FIFO. It then sends the pixel data down the FIFO. The appropriate number of long words is sent for each line and the host buffer pointer is adjusted to take into account the case where the same long word contains the last pixel of one line and the first pixel of another line. In this case the same long word will be sent down the FIFO twice. The microcode uses the dma_line, margin and haddr parameters to calculate how many words are used by each line. It then does a GE5 Data RAM to RE2 DMA transfer of the appropriate number of words in each line.

The destination bitplanes are selected with the GE_RWMODE token. The pixel data writes are subject to the screen mask clipping, WID checking and pattern mask checking if they are enabled. The pixel data can also have a raster operation applied to them as they are written. If pixel packing is used then the raster operation must be equal to 3 (SRC_COPY). If pixel packing is not used then the raster operation can be any of the 16 legal values specified with the GE_RASTEROP token. Refer to the Raster Subsystem chapter for specific details on these subjects.

Data Parameters :

data type	parameter name	parameter description
int	dma_line	specifies the long word count for a long word aligned line
int	dma_total	specifies the total number of word to be sent down the FIFO
int	upacmode	specifies the number of pixels/long -1
int	x1	x location of lower left corner
int	xlen	x direction pixel count
int	y1	y location of lower left corner
int	ylen	specifies the number of rows in rectangle
int	margin	flag used by the microcode to calculate the actual word counts for each line
int	haddr[0]	row 0 first pixel offset
int	haddr[1]	row 1 first pixel offset
int	haddr[2]	row 2 first pixel offset
int	haddr[3]	row 3 first pixel offset
int	pixel_data	pixel data to be written in rectangle

Return Data from Microcode Data RAM :

address	description
GEPIXELS	indicates if the pixel data area needs to be saved during a context switch

Example Usage :

```
/* Example for sending a buffer packed with shorts */

#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"

im_GEsetup;

short     pix_buffer[200];
long      pixsize = sizeof(short);
long      dma_line, dma_total, i, j;
long      upacmode = pixsize - 1;
long       x1 = 200, xlen = 20, y1 = 300, ylen= 30:
long      tmp, offset, offlong, margin, haddr[4], dma[4], *lptr;

tmp = (long) &pix_buffer[0];          /* convert buffer address to long */
lptr = (long *) (tmp & 0xFFFFFFFC);   /* get long word buffer boundary */
offset = tmp & 3;                     /* get buffer offset from long word boundary */

/* get offset of last byte of line from a long word boundary if first byte started on a long word
boundary. */

offlong = (xlen * pixsize) & 3;

/* Calculate the number of long words each line requires if it started on a long word boundary
* /

dma_line = (xlen * pixsize + sizeof(long) -1) >>2;

/* Calculate the first pixel offsets for each of the four lines. The first is just the offset of the
first pixel in the first word. The other three are the combination of first pixel offset of the
previous line plus the amount each line goes past the last long modulo 4. */

haddr[0] = offset;
haddr[1] = haddr[0] + offlong & 3;
haddr[2] = haddr[1] + offlong & 3;
haddr[3] = haddr[2] + offlong & 3;

/* calculate the margin flag. This represents the number of bytes between the end of the last
byte of the row and the next long word boundary. */

margin = 4 - offlong & 3;
```

```
/* Calculate the actual dma word counts for the four lines. This is the long word aligned dma
count which is incremented if the starting offset is > the margin. In this case it means that the
offset of the first byte will cause the last byte of the line to cross an additional long word
boundary. */

for (i = 0 ; i < 4 ; i++)
    dma[i] = dma_line + ((haddr[i]  > margin) ? 1 : 0;

/* Calculate the total number of words to be sent down the FIFO. This total must be <= 1280.
This is done by calculating the dma total for the full groups of four lines and then adding in the
lengths for the remaining partial group of four lines. */

dma_total = (dma[0] + dma[1] + dma[2] + dma[3]) * (ylen >> 2);
for (i = 0; i < (ylen & 3) ; i++)
    dma_total += dma[i];

/* For the cases where the margin value is never going to cause the dma counts of the four lines
to be different from the dma_line value set margin to -1 so the microcode does not do any extra
work. These cases are when the line starts and ends on a long word boundary or when the
margin is 3 which means haddr can never be > it. */

if (margin ==3 || (offset == 0 && offlong == 0))
    margin = -1;

GEWAIT;
GE[GE_RECTWRITE].i = 0;
GE[GE_DATA].i = dma_line;
GE[GE_DATA].i = dma_total;
GE[GE_DATA].i = upacmode;
GE[GE_DATA].i = x1;
GE[GE_DATA].i = xlen;
GE[GE_DATA].i = y1;
GE[GE_DATA].i = ylen;
GE[GE_DATA].i = margin;
GE[GE_DATA].i = haddr[0];
GE[GE_DATA].i = haddr[1];
GE[GE_DATA].i = haddr[2];
GE[GE_DATA].i = haddr[3];

/* Now send the pixel data down the FIFO. If the haddr of the next line is nonzero then
decrement the long buffer ptr to send the current word again. This is the case where the last
pixel of the current line and the first pixel of the next line are in the same word. */

for (i = 0 ; i < ylen ; i++) {
     for (j = 0 ; j < dma[i&3]; j++)
      GE[GE_DATA].i = *lptr++;          /* send long words */
      if (haddr[(i+1)&3])
       lptr--;
}

/* Turn off GEPIXELS flag so pixel area is not saved by next context switch. */

GE[GE_LOADGE].i = 0;
```

```
GE[GE_DATA].i = GEPIXELS;
GE[GE_DATA].i = -1;
```

GE_RESETLS

This token is used to enable or disable the use of continuous line stipple patterns for line segment drawings.

Data Parameters :

data type	parameter name	parameter description
int	ls_flag	line stipple enable/disable flag -1 disable line stipple 1 enable line stipple

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     ls_flag = 1;

GEWAIT;
GE[GE_RESETLS].i = 0;
GE[GE_DATA].i = ls_flag;
```

GE_RGBCOLOR

This token is used to set a color in RGB mode. The token is followed by the red, green and blue color values.

Data Parameters :

data type	parameter name	parameter description
int	red	Red color value (0 - FF hex)
int	green	Green color value (0 - FF hex)
int	blue	Blue color value (0 - FF hex)

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     red = 0xFF, green = 0xFF, blue = 0;  /* Yellow */

GEWAIT;
GE[GE_RGBCOLOR].i = 0;
GE[GE_DATA].i = red;
GE[GE_DATA].i = green;
GE[GE_DATA].i = blue;
```

GE_RGBSHADERANGE

This token is used to set RGB mode depth cueing parameters. The RGB intensity range is specified for each color as well as the z depth range. The RGB intensity values are inversely mapped to the z depth range. The less depth a pixel has the higher the RGB intensity it will be assigned.

Data Parameters :

data type	parameter name	parameter description
int	z1	low z value of z depth range
int	z2	high z value of z depth range
int	rmin	minimum red intensity value
int	rmax	maximum red intensity value
int	gmin	minimum green intensity value
int	gmax	maximum green intensity value
int	bmin	minimum blue intensity value
int	bmax	maximum blue intensity value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     z1 = 5, z2 = 40;
int     rmin = 7, rmax = 423;
int     gmin = 0, gmax = 1000;
int     bmin = 600, bmax = 2000;

GEWAIT;
GE[GE_RGBSHADERANGE].i = 0;
GE[GE_DATA].i = z1;
GE[GE_DATA].i = z2;
GE[GE_DATA].i = rmin;
GE[GE_DATA].i = rmax;
GE[GE_DATA].i = gmin;
GE[GE_DATA].i = gmax;
GE[GE_DATA].i = bmin;
GE[GE_DATA].i = bmax;
```

GE_RMOVE2

This token is used to update the current graphics drawing position. The 2-D position is specified in relative world coordinates and the coordinate data parameters are specified using floating point format.

Data Parameters :

data type	parameter name	parameter description
float	mx	relative x graphics position
float	my	relative y graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float     mx = 4.0, my = 2.0;

GEWAIT;
GE[GE_RMOVE2].i = 0;
GE[GE_DATA].f = mx;
GE[GE_DATA].f = my;
```

GE_RMOVE2I

This token is used to update the current graphics drawing position. The 2-D position is specified in relative world coordinates and the coordinate data parameters are specified using integer format.

Data Parameters :

data type	parameter name	parameter description
int	mx	relative x graphics position
int	my	relative y graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int         mx = 4, my = 2;

GEWAIT;
GE[GE_RMOVE2I].i = 0;
GE[GE_DATA].i = mx;
GE[GE_DATA].i = my;
```

GE_RMOVE3

This token is used to update the current graphics drawing position. The 3-D position is specified in relative world coordinates and the coordinate data parameters are specified using floating point format.

Data Parameters :

data type	parameter name	parameter description
float	mx	relative x graphics position
float	my	relative y graphics position
float	mz	relative z graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float     mx = 4.0, my = 2.0, mz = 6.0;

GEWAIT;
GE[GE_RMOVE3].i = 0;
GE[GE_DATA].f = mx;
GE[GE_DATA].f = my;
GE[GE_DATA].f = mz;
```

GE_RMOVE3I

This token is used to update the current graphics drawing position. The 3-D position is specified in relative world coordinates and the coordinate data parameters are specified using integer format.

Data Parameters :

data type	parameter name	parameter description
int	mx	relative x graphics position
int	my	relative y graphics position
int	mz	relative z graphics position

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     mx = 4, my = 2, mz = 6;

GEWAIT;
GE[GE_RMOVE3I].i = 0;
GE[GE_DATA].i = mx;
GE[GE_DATA].i = my;
GE[GE_DATA].i = mz;
```

GE_RVERTEX2

This token is used to send a relative 2-D vertex to the microcode. The x, y parameters are world coordinates and are in floating point format.

Data Parameters :

data type	parameter name	parameter description
float	vx	relative vertex x value
float	vy	relative vertex y value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float       vx = 10.0, vy = 4.0;

GEWAIT;
GE[GE_RVERTEX2].i = 0;
GE[GE_DATA].f = vx;
GE[GE_DATA].f = vy;
```

GE_RVERTEX2I

This token is used to send a relative 2-D vertex to the microcode. The x, y parameters are world coordinates and are in integer format.

Data Parameters :

data type	parameter name	parameter description
int	vx	relative vertex x value
int	vy	relative vertex y value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int         vx = 5, vy = 20;

GEWAIT;
GE[GE_RVERTEX2I].i = 0;
GE[GE_DATA].i = vx;
GE[GE_DATA].i = vy;
```

GE_RVERTEX3

This token is used to send a relative 3-D vertex to the microcode. The x, y, z parameters are world coordinates and are in floating point format.

Data Parameters :

data type	parameter name	parameter description
float	vx	relative vertex x value
float	vy	relative vertex y value
float	vz	relative vertex z value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "geomds.h"

im_GEsetup;

float       vx = 10.0, vy = 4.0, vz = 27.0;

GEWAIT;
GE[GE_RVERTEX3].i = 0;
GE[GE_DATA].f = vx;
GE[GE_DATA].f = vy;
GE[GE_DATA].f = vz;
```

GE_RVERTEX3I

This token is used to send a relative 3-D vertex to the microcode. The x, y, z parameters are world coordinates and are in integer format.

Data Parameters :

data type	parameter name	parameter description
int	vx	relative vertex x value
int	vy	relative vertex y value
int	vz	relative vertex z value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int          vx = 5, vy = 20, vz = 1;

GEWAIT;
GE[GE_RVERTEX3I].i = 0;
GE[GE_DATA].i = vx;
GE[GE_DATA].i = vy;
GE[GE_DATA].i = vz;
```

GE_RWMODE

This token is used to set the destination bitplanes for pixel writes. The bitplanes which can be written include the frame buffer bitplanes, the pup bitplanes, the uaux bitplanes, the window ID bitplanes and the Z buffer bitplanes as well as combinations of these as shown below. The GE_RWMODE token selects the destination bitplanes for the GE_WRITEPIXELS, the GE_WRITEBLOCK, the GE_RECTWRITE and the GE_RECTCOPY tokens.

Data Parameters :

data type	parameter name	parameter description
int	rwmode	desired read source 0 - Frame buffer bitplanes 1 - PUP bitplanes 2 - UAUX bitplanes 3 - Z buffer bitplanes 4 - Window ID (WID) bitplanes 5 - invalid 6 - Frame buffer port 7 - Z buffer port

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     rwmode = 0;   /* frame buffer bitplane access */

GEWAIT;
GE[GE_RWMODE].i = 0;
GE[GE_DATA].i = rwmode;
```

GE_SBOX

This token is not currently implemented by the GE5 microcode and should not be used. The GE_MOVE2 and GE_DRAW2 tokens should be used instead to draw the desired rectangles.

GE_SBOXF

This token is used to draw a screen aligned filled rectangle. The location of the rectangle is defined as the x, y screen locations of two opposite corners. The rectangle which is drawn cannot be rotated. The screen locations are world coordinates and are in a floating point format.

Data Parameters :

data type	parameter name	parameter description
float	x1	x coordinate of a corner of the box
float	y1	y coordinate of a corner of the box
float	x2	x coordinate of opposite corner of box
float	y2	y coordinate of opposite corner of box

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include  "mgr.h"
#include  "imsetup.h"
#include  "gecmds.h"

im_GEsetup;

float       x1 = 100.0, y1 = 200.0, x2 = 400.0, y2 = 800.0;

GEWAIT;
GE[GE_SBOXF].i = 0;
GE[GE_DATA].f = x1;
GE[GE_DATA].f = y1;
GE[GE_DATA].f = x2;
GE[GE_DATA].f = y2;
```

GE_SBOXFI

This token is used to draw a screen aligned filled rectangle. The location of the rectangle is defined as the x, y screen locations of two opposite corners. The rectangle which is drawn cannot be rotated. The screen locations are world coordinates and are in a integer format.

Data Parameters :

data type	parameter name	parameter description
int	x1	x coordinate of a corner of the box
int	y1	y coordinate of a corner of the box
int	x2	x coordinate of opposite corner of box
int	y2	y coordinate of opposite corner of box

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int         x1 = 100, y1 = 200, x2 = 400, y2 = 800;

GEWAIT;
GE[GE_SBOXFI].i = 0;
GE[GE_DATA].i = x1;
GE[GE_DATA].i = y1;
GE[GE_DATA].i = x2;
GE[GE_DATA].i = y2;
```

GE_SBOXI

This token is not currently implemented by the GE5 microcode and should not be used. The GE_MOVE2I and GE_DRAW2I tokens should be used instead to draw the desired rectangles.

GE_SCREENCLEAR

This token is used to clear the bitplanes selected with the GE_DRAWMODE token to the current color which would normally be BLACK but can be any valid color. If the PUP, UAUX or WID bitplanes are being cleared then the current color is set with either the GE_COLOR or GE_COLORF tokens. If the Frame Buffer bitplanes are being cleared then the GE_PIXTYPE token determines the pixel type. If the pixel type is set for an RGB pixel then the GE_RGBCOLOR token specifies the current color. If the pixel type is color index then the GE_COLOR or GE_COLORF token specifies the current color.

The write mask for the bitplanes being cleared must be set appropriately to allow all the bits to be written with the current color. The GE_PIXWRITEMASK is used to specify the write mask for the Frame Buffer bitplanes. The GE_AUXWRITEMASK token is used to specify the write mask for the PUP, UAUX and WID bitplanes. Since the GE_DRAWMODE token only allows the various bitplanes to be accessed individually they must be cleared separately.

The apeed of the screen clear is determined by the type of RE2 instruction which is used to perform the pixel writes. If the window being cleared is composed of less than five pieces then the piece list specified by the GE_SETPIECES token is used to clip the pixel writes and the Draw Flat 4 Spans are used to clear the window. Since this instruction does not perform WID checking it is at least 10 times faster than the Shaded Span instruction which does the WID checking. If the window consists of five or more pieces then the Shaded Span instruction must be used to do WID checked pixel writes so only the appropriate pieces are written into. In this case the piece list will have only one valid piece which will be used as the outer boundary for the clear screen operation. The window manager host software uses the SIMPLE flag in the GE5 data RAM to control which of these operations are performed. The SIMPLE flag should be set to 1 for the case of 1 to 4 pieces and it should be set to -1 for the case of 5 or more pieces.

If the raster op is set to anything other than 3 by the GE_RASTEROP token then the Shaded Span instruction will be used to clear the screen with the corresponding performance degradation. The screen will be cleared with the result of a bitwise logical operation between the current color and the destination bitplanes.

The screen clear can be done with the pattern mask enabled. This will cause the pattern mask to be applied as the screen is cleared to the current color. Only those screen bits which have a corresponding pattern mask bit of one will be written. The Shaded Span instruction will be performed to allow the pattern mask to be applied with the corresponding performance degradation.

If the picking mode is enabled the screen will not be cleared and the MOREHITS flag in the GE5 data RAM will be set to 1.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
```

```
#include "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_SCREENCLEAR].i = 0;
```

GE_SCRMASK

This token is used to set the RE2 screen mask to the data parameters which follow the token. The data parameters specify a rectangle which is used by the RE2 to clip pixel writes. If the X or Y pixel location is outside the rectangular screen mask the pixel write is prevented. Refer to the Raster Subsystem chapter for additional details on the use of the hardware screen mask. The first data parameter is the clipflag which indicates if the screen mask rectangle is smaller than the viewport.

Data Parameters :

data type	parameter name	parameter description
int	clipflag	screen mask clipping flag -1 screen clip mask = viewport 1 screen clip mask < viewport
int	llx	screen mask lower left x location
int	lly	screen mask lower left y location
int	urx	screen mask upper right x location
int	ury	screen mask upper right y location

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "gl.h"
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     clipflag = -1;
int     llx = 0, lly = 0;
int     urx = XMAXSCREEN, ury = YMAXSCREEN;

GEWAIT;
GE[GE_SCRMASK].i = 0;  /* Set full screen clip mask */
GE[GE_DATA].i = clipflag;
GE[GE_DATA].i = llx;
GE[GE_DATA].i = lly;
GE[GE_DATA].i = urx;
GE[GE_DATA].i = ury;
```

GE_SETMATRIX

This token is used to load a matrix onto the special PHIGSs matrix stack. The PHIGSs stack occupies the bottom six matrix entries of the sixteen total matrix entries. The data parameters specify the matrix index of the matrix which is to be loaded and the 4 X 4 matrix data to be loaded. This token is not currently used by any of the graphics library calls.

Data Parameters :

data type	parameter name	parameter description
int	index	matrix index in PHIGSs stack
float	a11	matrix entry a sub 11
float	a12	matrix entry a sub 12
float	a13	matrix entry a sub 13
float	a14	matrix entry a sub 14
float	a21	matrix entry a sub 21
float	a22	matrix entry a sub 22
float	a23	matrix entry a sub 23
float	a24	matrix entry a sub 24
float	a31	matrix entry a sub 31
float	a32	matrix entry a sub 32
float	a33	matrix entry a sub 33
float	a34	matrix entry a sub 34
float	a41	matrix entry a sub 41
float	a42	matrix entry a sub 42
float	a43	matrix entry a sub 43
float	a44	matrix entry a sub 44

Return Data from Microcode Data RAM :

None

Example Usage :

```
/* Load identity matrix */

#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int        index = 11;

GEWAIT;
GE[GE_SETMATRIX].i = 0;
GE[GE_DATA].i = index;
GE[GE_DATA].f = 1.0;
GE[GE_DATA].f = 0.0;
GE[GE_DATA].f = 0.0;
GE[GE_DATA].f = 0.0;
```

```
GE[GE_DATA].f = 0.0;
GE[GE_DATA].f = 1.0;
GE[GE_DATA].f = 0.0;
GE[GE_DATA].f = 0.0;
GE[GE_DATA].f = 0.0;
GE[GE_DATA].f = 0.0;
GE[GE_DATA].f = 1.0;
GE[GE_DATA].f = 0.0;
GE[GE_DATA].f = 0.0;
GE[GE_DATA].f = 0.0;
GE[GE_DATA].f = 0.0;
GE[GE_DATA].f = 1.0;
```

GE_SETPATTERN

This token is used to set the current pattern bit masks. The pattern is used in drawing textured polygons. This token enables textured pattern drawing.

Data Parameters :

data type	parameter name	parameter description
int	pat_index	pattern index number
int	bit_mask	bit mask consisting of 64 shorts

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int       pat_index = 2, i;
short     bit_mask[64], *masks;

for (i = 0 ; i < 64 ; i++)
   bit_mask = 0xAAAA;  /* 32X32 checkerboard pattern */

GEWAIT;
GE[GE_SETPATTERN].i = 0;
GE[GE_DATA].i = pat_index;

masks = bit_mask;

for (i = 0 ; i < 64 ; i+=2) {
    GE[GE_DATA].i = masks[i+1];     /* get lower short */
    GE[GE_DATA].i = masks[i];  /* get upper short */
}
```

GE_SETPIECES

This token is used to specify the rectangular pieces which comprise a screen window. If the window is unobscured and is therefore a single piece then the number of pieces is set to one and the rectangle is set for the size of the rectangle surrounding the window. If the window is obscured and is divided into 2 to 4 pieces the piecelist is set accordingly. If the window is obscured and more than 4 pieces then the piecelist is set to one and the rectangle is set for the outer bounding rectangle of all the pieces. The piece list is only used by the GE_SCREENCLEAR token to provide a performance improvement. When the window is 1 to 4 pieces it can be cleared using the Flat Draw instructions of the RE2 which does not use the WID checking feature. This causes the screen clear to be an order of magnitude faster than when WID checking is performed. After the window becomes 5 or more pieces then the WID checking must be performed so the Shaded Span instruction is performed at the greatly reduced performance.

Even if only one piece is valid in the piece list the 16 parameters must still be sent down the FIFO.

Data Parameters :

data type	parameter name	parameter description
int	npieces	number of valid pieces
int	ylen1	piece 1 y length
int	lly1	piece 1 lower left y location
int	xlen1	piece 1 x length
int	llx1	piece 1 lower left x location
int	ylen2	piece 2 y length
int	lly2	piece 2 lower left y location
int	xlen2	piece 2 x length
int	llx2	piece 2 lower left x location
int	ylen3	piece 3 y length
int	lly3	piece 3 lower left y location
int	xlen3	piece 3 x length
int	llx3	piece 3 lower left x location
int	ylen4	piece 4 y length
int	lly4	piece 4 lower left y location
int	xlen4	piece 4 x length
int	llx4	piece 4 lower left x location

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int         npieces = 1, rect [16] = {YMAXSCREEN, 0, XMAXSCREEN, 0,
                                      0, 0, 0, 0,
```

```
                                    0, 0, 0, 0,
                                    0, 0, 0, 0 }

    GEWAIT;
    GE[GE_SETPIECES].i = 0;
    GE[GE_DATA].i = npieces;

    for (i = 0 ; i < 16 ; i++)
        GE[GE_DATA].i = pieces[i];
```

GE_SETSURFSCALE

This token is part of the NURBS support and is used to reparameterize each patch within the surface to [0, 1]. The four parameters specify the two parametric directions and scale values.

Data Parameters :

data type	parameter name	parameter description
float	so	first parametric surface direction offset
float	to	second parametric surface direction offset
float	ss	first parametric surface scale value
float	ts	second parametric surface scale value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float           so, to, ss, ts;

GEWAIT;
GE[GE_SETSURFSCALE].i = 0;
GE[GE_DATA].f = so;
GE[GE_DATA].f = to;
GE[GE_DATA].f = ss;
GE[GE_DATA].f = ts;
```

GE_SETV

This token is part of the NURBS support and is used to compute the reparameterized second parametric value used in the evaluation of the surface patch.

Data Parameters :

data type	parameter name	parameter description
float	v	one coord of a parameter-space point on the surface

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float          v;

GEWAIT;
GE[GE_SETV].i = 0;
GE[GE_DATA].f = v;
```

GE_SETVHI

This token is part of the NURBS support and is used to compute the DeCastelau coefficient vector for an incoming v value.

Data Parameters :

data type	parameter name	parameter description
float	v	one coord of a parameter-space point on the surface

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float          v;

GEWAIT;
GE[GE_SETVHI].i = 0;
GE[GE_DATA].f = v;
```

GE_SHADEMODEL

This token is used to select the shading model. GOURAUD shading causes a linear interpolation of the vertex colors to be placed in the interior pixels during fill operations. FLAT shading causes a single constant color to be used during fill operations.

Data Parameters :

data type	parameter name	parameter description
int	shademodel	the selected shade model value 1 - GOURAUD shading -1 - FLAT shading

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     shademodel = 1;

GEWAIT;
GE[GE_SHADEMODEL].i = 0;
GE[GE_DATA].i = shademodel; /* GOURAUD shading */
```

GE_SHADERANGE

This token is used to set color index mode depth cueing parameters. The color index intensity range is specified as well as the z depth range. The color index intensity values are inversely mapped to the z depth range. The less depth a pixel has the higher the color intensity it will be assigned.

Data Parameters :

data type	parameter name	parameter description
int	z1	low z value of z depth range
int	z2	high z value of z depth range
int	imin	minimum CI intensity value
int	imax	maximum CI intensity value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     z1 = 1, z2 = 50, imin = 7, imax = 423;

GEWAIT;
GE[GE_SHADERANGE].i = 0;
GE[GE_DATA].i = z1;
GE[GE_DATA].i = z2;
GE[GE_DATA].i = imin;
GE[GE_DATA].i = imax;
```

GE_SMOOTHPOINT

This token is used to enable or disable anti-aliasing in drawing points.

Data Parameters :

data type	parameter name	parameter description
int	sp_flag	smooth point anti-aliase flag -1 disable anti-aliasing for points 1 enable anti-aliasing for points

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int         sp_flag = 1;

GEWAIT;
GE[GE_SMOOTHPOINT].i = 0;
GE[GE_DATA].i = sp_flag;
```

GE_STRIP

This token is part of the NURBS support and is used to compute the object space vertices along a strip on the patch.

Data Parameters :

data type	parameter name	parameter description
float	Ub	the initial u-value
float	numsteps	number of steps in the u-direction
float	delta_u	the step size

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float          Uo, numsteps, delta_u;

GEWAIT;
GE[GE_STRIP].i = 0;
GE[GE_DATA].f = Uo;
GE[GE_DATA].f = numsteps;
GE[GE_DATA].f = delta_u;
```

GE_SUBPIXEL

This token is used to enable or disable the use of subpixel position calculations for anti-aliased lines.

Data Parameters :

data type	parameter name	parameter description
int	sp_flag	subpixel anti-aliase flag -1 disable subpixel anti-aliasing 1 enable subpixel anti-aliasing

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int         sp_flag = 1;

GEWAIT;
GE[GE_SUBPIXEL].i = 0;
GE[GE_DATA].i = sp_flag;
```

GE_SURFMODE

This token is part of the NURBS support and is used to indicate if a nurbs context exists.

Data Parameters :

data type	parameter name	parameter description
int	mode	nurbs context flag 0 - nurbs context does not exist 1 - nurbs context does exist

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int         mode = 0;

GEWAIT;
GE[GE_SURFMODE].i = 0;
GE[GE_DATA].i = mode;
```

GE_SURFNTURF

This token is part of the NURBS support and is used to indicate the polynomial order of the two parametrics. It also indicates if they are rational or non-rational.

Data Parameters :

data type	parameter name	parameter description
int	s_order	polynomial order of first parametric
int	t_order	polynomial order of second parametric
int	rational	flag indicating if surface is rational 0 - non-rational 1 - rational

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int       s_order, t_order, rational;

GEWAIT;
GE[GE_SURFNTURF].i = 0;
GE[GE_DATA].i = s_order;
GE[GE_DATA].i = t_order;
GE[GE_DATA].i = rational;
```

GE_SURFP1

This token is part of the NURBS support and is used to compute the reparameterized first parametric value used in the evaluation of the surface patch.

Data Parameters :

data type	parameter name	parameter description
float	u	one coord of a parameter-space point on the surface

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float          u;

GEWAIT;
GE[GE_SURFP1].i = 0;
GE[GE_DATA].f = u;
```

GE_SWAPMESH

This token is used to toggle the triangle mesh pointer. Two triangle mesh vertices are are stored. After each new triangle mesh vertex is specified the triangle comprising the new vertex and the two stored vertices is drawn. Then one of the stored vertices is replaced by the new vertex. The triangle mesh pointer is used to point to the stored vertex which is replaced by the new vertex. The GE_SWAPMESH token causes the pointer to toggle and point to the other vertex not currently being pointed at.

Data Parameters :

None

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include  "mgr.h"
#include  "imsetup.h"
#include  "gecmds.h"

im_GEsetup;

GEWAIT;
GE[GE_SWAPMESH].i = 0;
```

GE_VERSION

This token is used to read the microcode version number from the microcode data ram. This token causes the version number to be placed in the pick buffer pointed to by the address in the data ram variable PICKPTR. Since the version number would overwrite the first word of the pick data this token should not be sent while picking or feedback mode are active. The version number consists of a 32 bit value.

Data Parameters :

None

Return Data from Microcode Data RAM :

Address	Description
PICKPTR	address pointer to version number

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"
#include "ge5_glob.h"

im_GEsetup;

long        ptr, version;

GEWAIT;
GE[GE_VERSION].i = 0;

/* If a context switch occurs before the data is read it must account for the data to be read */

GL_HQM_WR(PICKPTR);        /* Set the HQ MAR to read pointer to version data */
DRAM_RD(PICKPTR, ptr);     /* Read the pointer */
GL_HQM_WR(ptr);            /* Set the HQ MAR to point to the version data */
DRAM_RD(ptr, version);     /* Read the version number */
```

GE_VERTEX2

This token is used to send an absolute 2-D vertex to the microcode. The x, y parameters are world coordinates and are in floating point format.

Data Parameters :

data type	parameter name	parameter description
float	vx	absolute vertex x value
float	vy	absolute vertex y value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float       vx = 10.0, vy = 4.0;

GEWAIT;
GE[GE_VERTEX2].i = 0;
GE[GE_DATA].f = vx;
GE[GE_DATA].f = vy;
```

GE_VERTEX2I

This token is used to send an absolute 2-D vertex to the microcode. The x, y parameters are world coordinates and are in integer format.

Data Parameters :

data type	parameter name	parameter description
int	vx	absolute vertex x value
int	vy	absolute vertex y value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int          vx = 5, vy = 20;

GEWAIT;
GE[GE_VERTEX2I].i = 0;
GE[GE_DATA].i = vx;
GE[GE_DATA].i = vy;
```

GE_VERTEX3

This token is used to send an absolute 3-D vertex to the microcode. The x, y, z parameters are world coordinates and are in floating point format.

Data Parameters :

data type	parameter name	parameter description
float	vx	absolute vertex x value
float	vy	absolute vertex y value
float	vz	absolute vertex z value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float       vx = 10.0, vy = 4.0, vz = 27.0;

GEWAIT;
GE[GE_VERTEX3].i = 0;
GE[GE_DATA].f = vx;
GE[GE_DATA].f = vy;
GE[GE_DATA].f = vz;
```

GE_VERTEX3I

This token is used to send an absolute 3-D vertex to the microcode. The x, y, z parameters are world coordinates and are in integer format.

Data Parameters :

data type	parameter name	parameter description
int	vx	absolute vertex x value
int	vy	absolute vertex y value
int	vz	absolute vertex z value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int         vx = 5, vy = 20, vz = 1;

GEWAIT;
GE[GE_VERTEX3I].i = 0;
GE[GE_DATA].i = vx;
GE[GE_DATA].i = vy;
GE[GE_DATA].i = vz;
```

GE_VERTEX4

This token is used to send an absolute 4-D homogeneous coordinate vertex to the microcode. The x, y, z, w parameters are in floating point format.

Data Parameters :

data type	parameter name	parameter description
float	vx	absolute vertex x value
float	vy	absolute vertex y value
float	vz	absolute vertex z value
float	vw	absolute vertex w value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float       vx = 10.0, vy = 4.0, vz = 27.0, vw = 1.0;

GEWAIT;
GE[GE_VERTEX4].i = 0;
GE[GE_DATA].f = vx;
GE[GE_DATA].f = vy;
GE[GE_DATA].f = vz;
GE[GE_DATA].f = vw;
```

GE_VERTEX4I

This token is used to send an absolute 4-D homogeneous coordinate vertex to the microcode. The x, y, z, w parameters are in integer format.

Data Parameters :

data type	parameter name	parameter description
int	vx	absolute vertex x value
int	vy	absolute vertex y value
int	vz	absolute vertex z value
int	vw	absolute vertex w value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int         vx = 5, vy = 20, vz = 3, vw = 1;

GEWAIT;
GE[GE_VERTEX4I].i = 0;
GE[GE_DATA].i = vx;
GE[GE_DATA].i = vy;
GE[GE_DATA].i = vz;
GE[GE_DATA].i = vw;
```

GE_WRITEBLOCK

This token is used to write a rectangular block of pixels from the host to the selected bitplanes in the Raster Subsystem. This token supports two different modes of operation which are block mode and line by line mode. The block mode is used when writing a rectangle which is not packed or zoomed. The line by line mode is used when the rectangle is in a packed format in the host buffer or the x or y zoom factor is greater than 1.0. The destination bitplanes are specified with the GE_RWMODE token. The normal use of the GE_WRITEBLOCK token is to write into the frame buffer bitplanes, however any of the bitplanes which can be selected with the GE_RWMODE token can be written into. The GE_ZOOMFACTOR token sets the x and y zoom factors. Refer to the section on Pixel DMA Support in the programming considerations section of the Geometry Subsystem chapter for a description of how the block and the line by line modes operate.

For the selected destination bitplanes the appropriate bitplane mask register will determine which bits in the bitplanes will actually be written. The GE_PIXWRITEMASK will affect which bits in the frame buffer would be written. The GE_AUXWRITEMASK token would control bit writes to the PUP, UAUX, WID and Z buffer bitplanes. The pixel values written to the frame buffer PUP or UAUX bitplanes can have the logical operation performed on each pixel before it is written. The GE_RASTEROP token is used to specify the desired logical operation to be performed.

The current screen mask will clip writes to any bitplanes to the bits on or within the screen mask rectangle. If window ID checking is enabled then the bits to be written will be WID checked and only those that pass the check will be written. The exception to the checking are the Z buffer and WID bitplanes. On the base configuration adapter the PUP bitplanes are also not WID checked. On the enhanced adapter the PUP planes will be WID checked.

The host buffer is expected to have data that is appropriately formatted for the destination bitplanes. Refer to the RE2 DMA support paragraphs in the Raster Subsystem chapter for a description of the pixel formats and the pixel packing requirements. When doing packed pixel transfers or x zooming of pixels the RE2 requires the raster operation specified by GE_RASTEROP to be set to 3 (SRC_COPY).

Data Parameters :

Block Mode

data type	parameter name	parameter description
int	upacmode	pixel packing mode for RE2 0 - 1 pixel/word
int	x	lower left x absolute screen coordinate of rect
int	xlen	x length of rectangle
int	y	lower left y absolute screen coordinate of rect
int	dma_flag	flag indicating if a GE interrupt is used to indicate the block dma transfer has completed. 1 - GE int used to signal block dma done -1 - GE int not used to signal dma done
int	ylen	y length of rectangle (this parameter may be repeated if the rectangle is in multiple blocks)
int	done_flag	flag which indicates the operation is done. -1 indicates dma done

Parameter Checks Performed by Microcode for Block Mode

x < -1280	Not checked and will produce undefined results
x < 0	Draw only the pixels which are located past 0 in the rectangle
x > 1280	No error condition reported and no pixels drawn into bitplanes
y < 0	Not checked and will produce undefined results
ylen < 1	First ylen must be >= 1 or finish 1 flag never gets set

Line By Line Mode

data type	parameter name	parameter description
int	upacmode	pixel packing mode for RE2 0 - 1 pixel/word 1 - 2 pixels/word 2 - undefined 3 - 4 pixels/word
int	x	lower left x absolute screen coordinate of rect
int	xlen	x length of rectangle in pixels
int	y	lower left y absolute screen coordinate of rect
int	ylen	y length of rectangle (this parameter is sent only once since each line is sent individually

The following parameters are sent only if upacmode is > 0

int	linesizlong	number of long words to be DMAed for each line
int	margin	flag used by ucode to calculate the actual number of words to transfer for each line
int	haddr0	pixel offset into first word of line 0
int	haddr1	pixel offset into first word of line 1
int	haddr2	pixel offset into first word of line 2
int	haddr3	pixel offset into first word of line 3

Parameter Checks Performed by Microcode for Line By Line Mode

x < -1280	Not checked and will produce undefined results
x < 0	Draw only the pixels which are located past >= 0 in the rectangle
x >= 1280	DMA_FLAG set to DMA_CANCEL and no pixels are drawn
y < 1	Not checked and will produce undefined results
ylen < 1	DMA_FLAG set to DMA_CANCEL and no pixels are drawn

Return Data from Microcode Data RAM :

address	description
DMA_FLAG	indicates if the microcode is ready to continue with the DMA transfers - 1 cancel the token execution and do no DMA operations 0 continue with token execution and do the DMA operations
HQMSAV	contains the last HQ MAR reg value saved with the GE_HQMSAV token

Example Usage :

/* Example 1 - Writing a rectangle from a buffer containing packed pixels and with x and y zoom factors > 1.0. Raster Op must be 3 when doing packed and/or x zoomed pixels. */

```
#include  "mgr.h"
#include  "imsetup.h"
#include  "gecmds.h"
#include  "ge5_glob.h"

im_GEsetup;

#define   ERROR              - 1          /* arbitrary, system dependent value */
#define   LNBYLN               4          /* arbitrary value matching do_pixel_dma */

long      token = GE_WRITEBLOCK;
int       x = 200, y = 400, xlen = 500, ylen = 20;
long      pixsize = 2;        /* each pixel occupies 2 bytes in the host buffer */
long      upacmode = 1;       /* 2 pixels/long word */
short     pixbuf[10000];
float     xzoom = 2.0, yzoom = 4.0;  /* parameters sent to GE_ZOOMFACTOR token */
int       flags = 0:

/* This example shows the writing of a packed zoomed rectangle using the do_pixel_dma
function described in the Pixel DMA Support paragraphs in the Programming Considerations
section of the Geometry Subsystem chapter. */

GE[GE_ZOOMFACTOR].i = 0;
GE[GE_DATA].f = xzoom;
GE[GE_DATA].f = yzoom;

if (upacmode > 0 || xzoom > 1.0 || yzoom > 1.0)
    flags |= LNBYLN;

if (do_pixel_dma (token, pixbuf, pixsize, flags, x, xlen, y, ylen) == ERROR)
    printf ("GE_WRITEBLOCK error!\n");
```

```
/* Example 2 - Writing a rectangle from a buffer containing unpacked pixels and with x and y
zoom factors = 1.0. Raster Op can be any legal value. */

long    token = GE_WRITEBLOCK;
int     x = 200, y = 400, xlen = 500, ylen = 20;
long    pixsize = 4;        /* each pixel occupies 4 bytes in the host buffer */
long    upacmode = 0;       /* 1 pixels/long word */
short   pixbuf[10000];
float   xzoom = 1.0, yzoom = 1.0;  /* parameters sent to GE_ZOOMFACTOR token */
int     flags = 0;

/* This example shows the writing of an unpacked unzoomed rectangle using the do_pixel_dma
function described in the Pixel DMA Support paragraphs in the Programming Considerations
section of the Geometry Subsystem chapter. */

GE[GE_ZOOMFACTOR].i = 0;
GE[GE_DATA].f = xzoom;
GE[GE_DATA].f = yzoom;

if (upacmode > 0 || xzoom > 1.0 || yzoom > 1.0)
    flags |= LNBYLN;

/* flags would still be just 0 so do_pixel_dma routine would do a block mode transfer */

if (do_pixel_dma (token, pixbuf, pixsize, flags, x, xlen, y, ylen) == ERROR)
    printf ("GE_WRITEBLOCK error!\n");
```

GE_WRITEPIXDMA

This token is no longer used since it was a subset of the GE_WRITEBLOCK tokens capabilities.

GE_WRITEPIXELS

This token is used to write a specified number of pixels, up to a maximum of 1280, along a single scan line to the bitplanes selected by the GE_RWMODE token. The normal usage of this token is to write Color Index or RGB pixels to the Frame Buffer bitplanes. The microcode uses the current character position as the starting x and y screen location for the pixel writes. This token supports pixel packing but not pixel zooming. The raster operation specified with the GE_RASTEROP token must be three (SRC_COPY) if pixel packing is used. The first parameter specifies the total number of pixels to be written. The second parameter specifies the packing mode which is the number of pixels/long -1. The third parameter specifies the first pixel offset in the first word. The fourth parameter specifies the number of pixel data words to be read from the FIFO and written into the GE5 data RAM. This count is also the dma count used by the HQ1 when it does the DMA transfer of the pixel data from the GE5 data RAM to the RE2 bitplanes.

After the host sends the token and the four data parameters down the FIFO it then sends the pixel data down the FIFO. The microcode reads the pixel data from the FIFO and stores it in the GE5 data RAM pixel buffer. Then the microcode performs a GE5 Data RAM to RE2 DMA transfer to the selected bitplanes. After the pixel data has been sent the host sends the GE_LOADGE token to tell the microcode to set the GEPIXELS flag to -1 indicating that the pixel buffer is empty.

Data Parameters :

data type	parameter name	parameter description
int	pix_count	the number of pixels to be written
int	upacmode	the number of pixels per long word
int	haddr	the offset of the first pixel in the first word
int	dma_count	the number of words to read from the FIFO
int	pix_data	the dma_count number of pixel data words to be written

Data Parameter Checks Performed by the Microcode

dma_count < 1	The token is aborted and no pixels are written
cpos x < 0	Only pixels for x >= 0 will be displayed
cpos x >= 1279	The token is aborted and no pixels are written
cpos x + pix_count > viewport right	The char position is made invalid after the pixels are written
cpos invalid	If cpos is invalid then the token is aborted and no pixels are written cpos is invalid if cpos x > right edge of the screen or if cpos x > viewport right edge on previous token execution

Return Data from Microcode Data RAM :

address	description
GEPIXELS	indicates if the pixel data area needs to be saved during a context switch

Example Usage :

```
/* Example 1 - Unpacked pixel transfer.  Raster Op can be any of the 16 legal values */
```

```
#include  "mgr.h"
#include  "imsetup.h"
#include  "gecmds.h"
#include  "ge5_glob.h"

im_GEsetup;

long        pix_count = 250;
long        pix_buffer[250];
long        upacmode = 0, haddr = 0;

GEWAIT;
GE[GE_WRITEPIXELS].i = 0;
GE[GE_DATA].i = pix_count;
GE[GE_DATA].i = upacmode;
GE[GE_DATA].i = haddr;
GE[GE_DATA].i = pix_count;

for (i = 0 ; i < pix_count ; i++)
    GE[GE_DATA].i = pix_buffer[i];

/* Turn off GEPIXELS flag so pixel area is not saved by next context switch. */

GEWAIT;
GE[GE_LOADGE].i = 0;
GE[GE_DATA].i = GEPIXELS;
GE[GE_DATA].i = -1;

/* Example 2 - Packed pixel transfer.  Raster Op must be 3. */

#include  "mgr.h"
#include  "gecmds.h"

long        pix_count = 250;
short       pix_buffer[250];
short       pix_size = sizeof (short);   /* get pixel byte count for buffer data type */
long        tmp = (long) &pix_buffer[0];
long        *lptr = (long *) tmp & 0xFFFFFFFC;       /* get long word address of buffer */
long        upacmode = pix_size - 1;
long        haddr = tmp & 3;                /* get the buffer offset from a long */
long        dma_count = (pix_count * pix_size + (tmp & 3) + sizeof (long) - 1) >> 2;

GEWAIT;
GE[GE_WRITEPIXELS].i = 0;
GE[GE_DATA].i = pix_count;
GE[GE_DATA].i = upacmode;
GE[GE_DATA].i = haddr;
GE[GE_DATA].i = dma_count;

for (i = 0 ; i < dma_count ; i++)
    GE[GE_DATA].i = lptr++;             /* Must send long words */

/* Turn off GEPIXELS flag so pixel area is not saved by next context switch. */
```

```
GE[GE_LOADGE].i = 0;
GE[GE_DATA].i = GEPIXELS;
GE[GE_DATA].i = -1;
```

GE_ZBUFFER

This token is used to enable and disable Z buffer mode. When the GE_ZBUFFER token is sent to the microcode if Z buffer mode is being disabled then the Z Buffer mode will be turned off and the DEPTHFN and COLORCMP registers in the RE2 will be set to the default values described later. If Z Buffer mode is being enabled then the microcode will load the DEPTHFN registers with the z function value previously specified by the GE_ZFUNCTION token. The COLORCMP register will also be set with the value previously specified by the GE_ZSOURCE token. The register settings depend on the z source value.

If the z source is set for the Frame Buffer color as the source the register settings depend on whether the antialiase mode is enabled. If the antialiase mode is enabled the microcode will set the COLORCMP register for the color source and will set the DEPTHFN register to the value specified by the GE_ZFUNCTION token.. If the antialiase mode is not enabled then the COLORCMP and the DEPTHFN registers are set to the default values. The default value for the COLORCMP register is the Z Buffer source. The default value for the DEPTHFN register is for the compare to always pass. The fast Z clear enable bit is set for the enhanced configuration and is cleared for the base configuration.

If the z source is the Z Buffer the register settings depend on whether the optional Z Buffer card is installed. If the Z Buffer is installed the COLORCMP register is set for the Z Buffer as the source and the DEPTHFN register is set to the ZFUNCTION value specified by the GE_ZFUNCTION register. If the Z Buffer card is not installed then the registers are set to the default values specified above.

Refer to the Raster Subsystem chapter for additional information on the Z compare operation and the COLORCMP and DEPTHFN registers.

Data Parameters :

data type	parameter name	parameter description
int	zflag	zbuffer enable/disable flag -1 disable zbuffer mode 0 enable zbuffer mode

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int         zflag = 0;

GEWAIT;
GE[GE_ZBUFFER].i = 0;
GE[GE_DATA].i = zflag;
```

GE_ZCLEAR

This token is never used and is not supported in the microcode. It is only listed here because it is in the gecmds.h file.

GE_ZFUNCTION

This token is used to specify the comparison function for the Z comparison circuitry in the RE2 when Z Buffer mode is enabled. The function controls the relational comparison performed between the source specified by the GE_ZSOURCE token and a new value. If the z source is the Z buffer the new Z values are compared with the Z Buffer values. If the z source is the Frame Buffer the new color values are compared with the Frame Buffer values. The result of the comparison is used to condition pixel writes. If the comparison passes then the pixel write is enabled and if the comparison fails then the pixel write is disabled.

The Z function is composed of four bits with the first three specifying the relational comparison function and the fourth bit controlling the enabling or disabling of the fast Z clear mode of operation. If bit 3 is set and the least significant bit in the Window ID is set then the source value is made invalid and the compare passes. The z function value is used to set the DEPTHFN register in the RE2 when Z Buffer mode is enabled.

The GE_DEPTHFN token can be used to temporarily override the Z function value so that the Z Buffer can be cleared with the fast Z clear method or with the GE_CZCLEAR token. After the clear is done the the desired Z function would be reloaded using the GE_ZFUNCTION token again.

When the GE_ZFUNCTION token is sent to the microcode, if the Z Buffer mode has not been enabled the z function value will be saved in the GE5 data RAM and will be used to set the DEPTHFN register when the Z Buffer mode is enabled using the GE_ZBUFFER token. If the Z buffer mode is currently enabled then the microcode will load the DEPTHFN registers based on the specified Z function value. The COLORCMP register will also be set with the value previously specified by the GE_ZSOURCE token. The register settings depend on the z source value.

If the z source is set for the Frame Buffer color as the source the register settings depend on whether the antialiase mode is enabled. If the antialiase mode is enabled the microcode will set the COLORCMP register for the color source and will set the DEPTHFN register to the value specified by the GE_ZFUNCTION token.. If the antialiase mode is not enabled then the COLORCMP and the DEPTHFN registers are set to the default values. The default value for the COLORCMP register is the Z Buffer source. The default value for the DEPTHFN register is for the Z compare to always pass which is the Z Buffer disabled mode setting. The fast Z clear enable bit is set for the enhanced configuration and is cleared for the base configuration.

If the z source is the Z Buffer the register settings depend on whether the optional Z Buffer card is installed. If the Z Buffer is installed the COLORCMP register is set for the Z Buffer as the source and the DEPTHFN register is set to the ZFUNCTION value specified by the GE_ZFUNCTION register. If the Z Buffer card is not installed then the registers are set to the default values specified above.

Refer to the Raster Subsystem chapter for additional information on the Z compare operation and the COLORCMP and DEPTHFN registers.

Data Parameters :

data type	parameter name	parameter description
int	zfunction	desired Z depth comparison function
	zfunction (3-0)	comparison operation

0000	never passes
0001	passes if src < dest
0010	passes if src = dest
0011	passes if src <= dest
0100	passes if src > dest
0101	passes if src <> dest
0110	passes if src >= dest
0111	always passes
1XXX	enable old Z invalidate if (WID LSB = 1)

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

int     zfunction = 1;      /* Z compare passes if < */

GEWAIT;
GE[GE_ZFUNCTION].i = 0;
GE[GE_DATA].i = zfunction;
```

GE_ZOOMFACTOR

This token is used to set the x and y zoom factors to specify the number of adjacent pixel locations which are written with each pixel value. The zoom factors are used by the GE_WRITEBLOCK and the GE_RECTCOPY tokens as they write pixels to the bitplanes. To cause no zoom to take place for these two tokens the x any zoom factors should be set to 1.0. The x and y zoom factors are initially set to 1.0 during the hard init after adapter reset. The GE_INIT token also resets the x and y zoom factors to 1.0. The zoom factors are specified as floats even though at this time they are converted to integer values before being used. If an x zoom greater than 1.0 is specified then the raster operation specified with the GE_RASTEROP token must be set to 3 (SRC_COPY).

Data Parameters :

data type	parameter name	parameter description
float	xzoom	x zoom factor
float	yzoom	y zoom factor

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float           xzoom = 10.0, yzoom = 3.0;

GEWAIT;
GE[GE_ZOOMFACTOR].i = 0;
GE[GE_DATA].f = xzoom;
GE[GE_DATA].f = yzoom;
```

GE_ZSOURCE

This token is used to specify the source for the Z comparison circuity in the RE2. The source for the Z comparator can be either the Z Buffer or the Frame Buffer. If the source is set for the Z Buffer the new z values are compared with the old z values from the Z buffer. If the source is set for the Frame Buffer the new color values are compared with the old color values from the Frame Buffer. The Z Buffer source is selected to condition pixel writes based on the distance of the pixel from the viewer which is the normal mode of operation. The Frame Buffer source is selected to improve the appearance of antialiased lines. If the Z Buffer mode has not been enabled the z source value will be saved in the GE5 data RAM and will be used to set the z source when the Z Buffer mode is enabled using the GE_ZBUFFER token. If the Z buffer mode is currently enabled then the microcode will load the COLORCMP and the DEPTHFN registers based on the specified z source.

If the z source is set for the Frame Buffer color as the source the register settings depend on whether the antialiase mode is enabled. If the antialiase mode is enabled the microcode will set the COLORCMP register for the color source and will set the DEPTHFN register to the value specified by the GE_ZFUNCTION token.. If the antialiase mode is not enabled then the COLORCMP and the DEPTHFN registers are set to the default values. The default value for the COLORCMP register is the Z Buffer source. The default value for the DEPTHFN register is for the compare to always pass. The fast Z clear enable bit is set for the enhanced configuration and is cleared for the base configuration.

If the z source is the Z Buffer the register settings depend on whether the optional Z Buffer card is installed. If the Z Buffer is installed the COLORCMP register is set for the Z Buffer as the source and the DEPTHFN register is set to the ZFUNCTION value specified by the GE_ZFUNCTION register. If the Z Buffer card is not installed then the registers are set to the default values specified above. Refer to the Raster Subsystem chapter for additional information on the Z compare operation and the COLORCMP and DEPTHFN registers.

Data Parameters :

data type	parameter name	parameter description
int	zsource	z comparison source flag -1 Frame Buffer source 1 Z Buffer source

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;
int       zsource = 1;  /* select the Z Buffer as the Z compare source */

GEWAIT;
GE[GE_ZSOURCE].i = 0;
GE[GE_DATA].i = zsource;
```

GE_1LOAD1

This token is part of the NURBS support and is used to load a single floating point value into the nurbs data area in the microcode data RAM.

Data Parameters :

data type	parameter name	parameter description
float	d0	floating point nurbs data value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup:

float           d0;

GEWAIT;
GE[GE_1LOAD1].i = 0;
GE[GE_DATA].f = d0;
```

GE_1LOAD3

This token is part of the NURBS support and is used to load three floating point values into the nurbs data area in the microcode data RAM.

Data Parameters :

data type	parameter name	parameter description
float	d0	floating point nurbs data value
float	d1	floating point nurbs data value
float	d2	floating point nurbs data value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include  "mgr.h"
#include  "imsetup.h"
#include  "gecmds.h"

im_GEsetup;

float             d0, d1, d2;

GEWAIT;
GE[GE_1LOAD3].i = 0;
GE[GE_DATA].f = d0;
GE[GE_DATA].f = d1;
GE[GE_DATA].f = d2;
```

GE_1LOAD4

This token is part of the NURBS support and is used to load four floating point values into the nurbs data area in the microcode data RAM.

Data Parameters :

data type	parameter name	parameter description
float	d0	floating point nurbs data value
float	d1	floating point nurbs data value
float	d2	floating point nurbs data value
float	d3	floating point nurbs data value

Return Data from Microcode Data RAM :

None

Example Usage :

```
#include "mgr.h"
#include "imsetup.h"
#include "gecmds.h"

im_GEsetup;

float           d0, d1, d2, d3;

GEWAIT;
GE[GE_1LOAD4].i = 0;
GE[GE_DATA].f = d0;
GE[GE_DATA].f = d1;
GE[GE_DATA].f = d2;
GE[GE_DATA].f = d3;
```

Microcode Data RAM Definitions

The following list describes the various symbolic address names defined in the ge5_glob.h file for the locations in the microcode data RAM. These addresses are used by the host to access data which represent the results of token executions or contain other miscellaneous data that the host needs to access.

Address	Description
BBOX_VISABLE	indicates if the bbox is pruned or culled
CB	current blue color value
CG	current green color value
CHARVALID	valid position flag (< 0 means invalid)
CPOSX	current Character Position X value
CPOSY	current Character Position Y value
CR	current color index or red color value
CURMATRIX	contains pointer to the current matrix
DMA_COUNT	contains the DMA context switch word count
DMA_FLAG	Indicates if a DMA operation needs to be done
GEPIXELS	indicates if the pixel data area needs to be saved during a context switch
GW	current graphics drawing w position
GX	current graphics drawing x position
GY	current graphics drawing y position
GZ	current graphics drawing z position
HQMSAV	contains contents of HQ MAR register
INTERRUPT	contains the GE interrupt type
NORMALMATRIX	contains a pointer to current normal matrix
NUMPIX	the number of pixels actually read
PICKPTR	address pointer to picking or feedback data
PICKSPACE	size of the picking or feedback data
PIXELMEM	pixel data in microcode data RAM
PUPDATA	contains the current PUP draw value
REVERSION	contains the microcode version number
UAUXDATA	contains the current UAUX draw value
WIDDATA	contains the current WID draw value