DMPSX version 3.01

Release History

Changes from Version 3.00

1) The bug that caused a hung up on the specific object has been fixed.
2) The file specifications with relative path has been enabled.

The "dmpsx.exe" is the only file changed from Version 3.00.

Changes from the previous versions(Version 2.XX)

1) Disclosed Load/Store instructions of the GTE register.
2) Supplied 3 new header files.
3) Supported assembler programs.
4) Deleted " inline.tbl".

"DMPSX version 3.0" and header files cannot be used for old versions of the DMPSX, nor old version of " inline.h" cannot be used for "DMPSX version 3.0" vice versa.
To use the program created on the old versions of the DMPSX on "DMPSX version 3.0", modify the program to include "   inline_o.h" or "inline_c.h" instead of " inline.h".

Changes from prevous versions prior to DMPSX version 2.06 (May 15, 1996)

1) This version corresponds to the overlaid text section (named "*.text").
2) Some bugs were fixed.
    ** changed description "::" in   inline.h to ": :" to avoid C++ compile errors
    ** fixed an error which was occurred by dividing text section
3) Some functions were added into ' inline.h' and ' inline.tbl'.

Added following functions

gte_ldrgb3c
gte_ldbkdir
gte_ldfcdir
gte_ldsvrtrow0
gte_ldsvllrow0
gte_ldsvlcrow0
gte_ldtr

gte_rtv0tr
gte_rtv1tr
gte_rtv2tr
gte_rtirtr
gte_rtv0bk
gte_rtv1bk
gte_rtv2bk
gte_rtirbk
gte_rtv0fc
gte_rtv1fc
gte_rtv2fc
gte_rtirfc

gte_llv0
gte_llv1
gte_llv2
gte_llir
gte_llv0tr
gte_llv1tr
gte_llv2tr
gte_llirtr
gte_llv0bk

gte_llv1bk
gte_llv2bk
gte_llirbk
gte_llv0fc
gte_llv1fc
gte_llv2fc
gte_llirfc

gte_lcv0
gte_lcv1
gte_lcv2
gte_lcir
gte_lcv0tr
gte_lcv1tr
gte_lcv2tr
gte_lcirtr
gte_lcv0bk
gte_lcv1bk
gte_lcv2bk
gte_lcirbk
gte_lcv0fc
gte_lcv1fc
gte_lcv2fc
gte_lcirfc

gte_stlvnl0
gte_stlvnl1
gte_stlvnl2
gte_mvlvtr

What is DMPSX?

DMPSX provides the programming environment for high-speed execution of the program related to the GTE processing.
The overhead upon function invocation can be eliminated by controlling GTE by including DMPSX unique header files into the program, then using the inline functions (macros) defined in the include files.

Moreover, no functions are called, instruction cache is easily controlled, and thus high-speed routines can be implemented.

The compile procedures for the DMPSX programs differ from that of the standard program. Compile a program to generate an object file "*.obj", then use this object file as an input to "dmpsx.exe". Finally link this file with other object files. Please refer to the later section "How to Compile" for detail.

Component Files

dmpsx.exe	Executable file. Processes the compiled object file.
inline_o.h	Header file to generate the same code as the old version DMPSX.
inline_c.h	Header file where register load/store instructions in the "inline_o.h" are optimized using inline asm feature.
inline_a.h	Converted "inline_c.h" file for assembler programs.
gtereg.h	GTE register macros are defined for assembler programs.
gtemac.h	Header file to implement the same "libgte.lib" functions in the DMPSX inline functions.

Syntax

```
Syntax:         DMPSX  object-file [ -o output-file ] [ -b ]

Options:        -o:     Specify the output file name
                        Default: override the input file
                -b:     Disable creation of the . bak (backup) file.


Example:        DOS> dmpsx main.obj
```

Overview

This program implements the GTE inline functions as described later. First, compile a source code with the GTE specified header files. Second, use the generated object file as the input of the DMPSX. Then GTE code will be expanded in the object file.

The inline functions having the same interface as ordinary function can be used by following the steps above.

(Note) The inline functions here has the same property as the ones in the "inline" function in C language, but are generated differently.

Inline Functions

When a program calls the libgte low level functions, such as RotTransPers, an instruction cache miss occurs, and thus the processing speed may decrease.
Using the inline functions can avoid this situation, and also execution time becomes faster.

```
*********************************************************   *********
*   Please note when an original program runs on cache, using    *
*   inline functions may delay the operation in stead.           *
******************************************************************
```

GTE inline function name has the following naming conventions.

```
        gte_******
```

GTE inline functions can be grouped roughly in two.
One is the basic functions for using GTE, the other is the functions that can be replaced with the libgte low level functions just as they are.
When the first letter following "gte_" is a lower case letter, it is a basic function. When it is a capital letter, on the other hand, it is a function that can be replaced with the libgte low level function as they are.

Example:

```
        gte_rtps                ...Basic function
        gte_RotTransPers        ...Replaceable function
```

All basic functions and some of the replaceable functions can directly become an object code. Most of the remaining replaceable functions are defined in "gtemac.h" as the basic macros.

Changing to Replaceable Functions

Follow the steps below to modify the normal function calls to inline function calls.

0. Include following header files

```
"inline_o.h" or " inline_c.h"
"gtemac.h"
```

1. Check whether the function in question is replaceable by searching the function name in " function.txt" or macro.txt1.

   <If exists>

   2. Make the function to an inline function by prefixing "gte_" to the function name.

```
(e.g.)  RotTransPers() ->  gte_RotTransPers()
```

   3. When a function has a return value, add   the return value pointer at the end of the argument list.

```
(e.g.)  otz=RotTransPers(...) ->  gte_RotTransPers(...,& otz)
```

   4. When a replaced inline function destroys GTE constants, such as Rotation Matrix, Transfer vector, etc., these constants need to be saved and loaded.

```
(e.g.)  OuterProduct12() ->  gte_ReadRotMatrix(&m)
                            gte_OuterProduct12()
                            gte_SetRotMatrix(&m)

    CompMatrix() ->        gte_ReadRotMatrix(&m)
                   gte_CompMatrix()
                   gte_SetTransMatrix(&m)
```

   These are the only two constants that will be destroyed and are different from the   libgte version.

   <If not exist>

   2. Write inline functions directly into the program.

```
(e.g.)  RotTransPers4() ->  gte_RotTransPers3()
                           gte_RotTransPers()
```

Inline Programming by Basic Functions

When no replaceable functions can be found or more
optimization is required, write basic functions directly
into the source program.
GTE normally operates in the following three steps.

```
1. Load input data      ... CPU Memory/Register -> GTE Register
2. Execute              ... GTE function execution
3. Store output data    ... GTE Register -> CPU Memory/Register
```

In the function list, " infunc.txt"

Type 1 : "Register Load Functions"
Type 2 : "GTE Commands"
Type 3 : "Register Store Functions"

For instance, in " gtemac.h" gte_RotTransPers can be written as follows.

```
gte_ldv0(r1);           /* type1:load 3D coordinate */
gte_rtps();             /* type2:Rotate,Transfer,Perspect */
gte_stsxy(r2);          /* type3:store 2D coordinate */
gte_stdp(r3);           /* type3:store depth  que p */
gte_stflg(r4);          /* type3:store flag */
gte_stszotz(r5);        /* type3:store  sz/4 as  otz */
```

If "depth queue p" and "flag" are not needed, they can be
omitted as follows.

```
        gte_ldv0(r1);           /* type1:load 3D coordinate */
        gte_rtps();             /* type2:Rotate,Transfer,Perspect */
        gte_stsxy(r2);          /* type3:store 2D coordinate */
        gte_stszotz(r5);        /* type3:store  sz/4 as  otz */
```

Moreover, executing a CPU process between the GTE command and the store functions, both GTE and CPU process will be executed concurrently until either one of them terminates.

```
        gte_ldv0(r1);           /* type1:load 3D coordinate */
        gte_rtps();             /* type2:Rotate,Transfer,Perspect */
        CPUprocess;             /* CPU process */
        gte_stsxy(r2);          /* type3:store 2D coordinate */
        gte_stszotz(r5);        /* type3:store  sz/4 as  otz */
```

However, inserting too many CPU process only makes the source code difficult to read, and thus is meaningless since the GTE command execution time is not long.

(NOTE)
The basic functions consist of the following three types.

Type 1 : "Register Load Functions"
Type 2 : "GTE Commands"
Type 3 : "Register Store Functions"

There is no guarantee in operation when Type 1 function is called between Type 2 and Type 3 as shown below.

Example of Incorrect Usage:

```
        gte_rtps();     /* type2:Rotate,Transfer,Perspect */
        gte_ldv0(r1);   /* type1:load 3D coordinate */
        gte_stsxy(r2);  /* type3:store 2D coordinate */
```

How to Use DMPSX for Assembler Programs

When using the DMPSX for Assembler programs, include "inline_a.h" header file. Within the "inline_a.h" file, only the type 2 instructions are defined. Please refer to the GTE documents for the direct GTE register operations. The GTE register macros are defined in "gtereg.h" file.

Following type 2 instructions within the "inline_a.h" require arguments.

```
        MVMVA   sf,mx,v,cv,lm
        SQR     sf
        OP      sf
        GPF     sf
        GPL     sf
```

Specify integers for all above arguments. Please refer to the GTE documents for the details of the commands and arguments.

How to set the environment

Files:

- dmpsx.exe
- inline_o.h
- inline_c.h
- inline_a.h
- gtereg.h
- gtemac.h

How to install:

1) Copy "dmpsx.exe" to the directory specified in "PATH" (e.g. "\psx\bin").

2) Copy "inline_o.h", "inline_c.h", "inline_a.h", "gtereg.h" and "gtemac.h" to the directory where the compiler include files exist (e.g. "\ psx\include").

How to Compile

Compile the DMPSX source code specifying the compile option "-c" to stop after generating an object file. Before linking, run "dmpsx.exe" using the object file in the previous step as an input. Then link this newly created object file with other object files.

(e.g.) For `ccpsx -O -Xo$80010000 use1.c use2.c a1.c a2.c -omain.cpe`
(where use1.c and use2.c use the DMPSX),

change the makefile as follows;

```
ccpsx -c -O  use1.c -ouse1.obj
dmpsx  use1.obj
ccpsx -c -O  use2.c -ouse2.obj
dmpsx  use2.obj
ccpsx -O -Xo$80010000  use1.obj  use2.obj  a1.c  a2.c -omain.cpe
```

Include File Description

** inline_o.h

In the header file "inline_o.h", types 1 and 3 instructions of the old version "inline.h" header file are disclosed to keep compatibility with previous versions.
The header file "inline_o.h" generates the same code as the old "inline.h", thus having the same degree of redundancies.
Especially when passing a C program data to the DMPSX, each inline includes one to four words of extra instructions.

** inline_c.h

In this header file, the redundancies in the "inline_o.h" stated above have been eliminated. Use this header file for standard program development.
Of those inline functions in the header file, type 1 instruction (GET Register Load Instruction) and type 3 instruction(GTE Register Store Instruction) are obsolete in the DMPSX from this version 3.0. Thus non-DMPSX program can also use these two instructions. The type 2 instruction (GET Command) need to be converted by "dmpsx.exe" before execution.
The header file "inline_c.h" holds the same function names and capabilities. But generates more efficient code as compared with "inline_o.h".

** inline_a.h

This is the header file for Assembler programs. Only the type 2 instructions (GTE commands) are defined within this header file.
Please refer to the GTE documents for the register specifications and the details of each instruction.

** gtereg.h

The GTE register macros are defined for assembler programs. The macro name for each register is the macro name prefixed with "C2_". Please refer to the GTE document for each register functions.

** gtemac.h

Being different from other files, this header file " gtemac.h" is not always required for the DMPSX.
In this header file, the same macro function names with the same capabilities provided in the " libgte.lib" are defined as combination of the basic CMPSX functions. Make use of this header file to start with in making a libgte program high-speed using the DMPSX.

DMPSX IN-LINE FUNCTION TABLE

0. Overview

This table describes;
- GTE registers available for GTE
- How in-line functions can be sorted

Please refer to the Reference Manual for the syntax detail of each function.

1. GTE Register

In this DMPSX document, GTE registers are described with the symbols below:

v0 : 3 dimensional short vector 0 for vertex coordinates or a normal vector
v1 : 3 dimensional short vector 1 for vertex coordinates or a normal vector
v2 : 3 dimensional short vector 2 for vertex coordinates or a normal vector

RGBcd(=rgb) : 4 dimensional character vector for R,G,B and G PU codes

sv(=ir) : General 3 dimensional short vector
lv : General 3 dimensional long vector

dp : Short scalar for depth queuing•iinterpolation•j

sxy0 : Short vector 0 for screen XY coordinates
Last word of the 3 WORD FIFO
sxy1 : Short vector 1 for screen XY coordinates
2nd word of 3 WORD FIFO
sxy2 : Short vector 2 for screen XY coordinates
1st word of 3 WORD FIFO

sz0 : Short scalar 0 for screen XY coordinates
Last word of 4 WORD FIFO
sz1 : Short scalar 1 for screen X Y coordinates
3rd word of 4 WORD FIFO
sz2 : Short scalar 2 for screen XY coordinates
2nd word of 4 WORD FIFO
sz3 : Short scalar 3 for screen XY coordinates
1st word of 4 WORD FIFO

otz : Short scalar for OTZ

opz : Long scalar for outer products

rgb3 : 4 dimensional character vector for R,G,B and GPU codes
3 WORD FIFO (rgb0,rgb1,rgb2)

lzc : Long scalar for Leading zero counter

BackColor(=bk) : 3 dimensional long vector for back color

FarColor(= fc) : 3 dimensional long vector for far color

Offset : 2 dimensional long vector for screen offset

Screen : Long scalar for the distance between view point and the screen

RotMatrix(= rt) : 3 X 3 Rotation matrix

LightMatrix(= ll): 3 X 3 Light source direction matrix

ColorMatrix(= lc): 3 X 3 Light source color matrix

Trans(= tr) : Translating 3 dimensional long vector

flg : Flag

2. Load Instructions

Instruction for loading a value to GTE

2.1. Load Instructions for Vertex Coordinates, Normal Line Vectors, etc.

gte_ldv0 : Load SVECTOR to vector v0.
gte_ldv1 : Load SVECTOR to vector v1.
gte_ldv2 : Load SVECTOR to vector v2.
gte_ldv3 : Load SVECTOR from non-continuous address to vectors v0, v1, and v2.
gte_ldv3c : Load SVECTOR from continuous address to vectors v0, v1, and v2.
gte_ldv3c_vertc : Load SVECTOR from vertex coordinate area of the structure •@VERTC defined in libgs.h.
gte_ldv01 : Load SVECTOR from non-continuous address to vectors v0,v1.
gte_ldv01c : Load SVECTOR from continuous address to vectors v0,v1.
gte_ldlv0 : Load lower 16 bits of VECTOR to vector v0.

2.2. Load Instructions for RGB, and GPU Code

gte_ldrgb : Load CVECTOR to vector rgb.
(GPU code is also overwritten.•j
gte_ldrgb3 : Load CVECTOR from non-continuous address to FIFO rgb0, rgb1, and rgb2.
The same value for rgb2 is loaded to rgb.
(For GPU setting)
gte_ldrgb3c : Load CVECTOR from continuous address to FIFO rgb0, rgb1, and rgb2.
The same value for the rgb2 is loaded to rgb.
(For GPU setting)
gte_SetRGBcd : =gte_ldrgb
Load CVECTOR to vector rgb.
(GPU code is also overwritten.)

2.3. Load Instructions for General Vectors

gte_ldlvl : Load lower 16 bits of VECTOR to general short vector.
gte_ldsv : Load SVECTOR to general short vector.
gte_ldbv : Load 2 dimensional byte vector to 1st and 2nd elements of general short vector.
gte_ldcv : Load R, G, and B of CVECTO to general short vector.
GPU code parts will not be loaded anywhere.
gte_ldclmv : Load the first column of MATRIX to general short vector.

2.4. Load Instructions for Depth Queuing Scalar Values

gte_lddp : Load scalar values for depth queuing.

2.5. Load Instructions for Screen Coordinates

gte_ldsxy0 : Load 1st vertex screen coordinates X Y.

gte_ldsxy1 : Load 2nd vertex screen coordinates X Y.

gte_ldsxy2 : Load 3rd vertex screen coordinates X Y.

gte_ldsxy3 : Load a value representing the screen coordinates X Y of 3 vertices.

gte_ldsxy3c : Load the screen coordinates X Y represented by pointers that locate on the continuous address.

gte_ldsz3 : Load the screen coordinates Z of 3 vertices.

gte_ldsz4 : Load the screen coordinates Z of 4 vertices.

2.6. Load Instructions for Vector for Outer Products Calculation

gte_ldopv1 : Load 1st vector in order to calculate the outer products. (3 X 3 rotation matrix is destroyed.)

gte_ldopv2 : Load 2nd vector in order to calculate the outer products. (General short vector is destroyed.)

2.7. Load Instructions for Values for LZC calculation

gte_ldlzc : Load a value in order to calculate LZC(Leading Zero Counter) Numbers of 0 or 1 following the MSB of the value loaded is calculated.

2.8. Load Instructions for Back Color Vector

gte_ldbkdir : Load long vector value to back color vector.

gte_SetBackColor : Load (RBK,GBK,BBK) values to back color vector. Each value is multiplied by 16 prior to its load.
•@For example, 256 will be 4096.

2.9. Load Instructions for Far Color Vector

gte_ldfcdir : Load long vector value to far color vector.

gte_SetFarColor : Load (RFC,GFC,BFC) values to far color vectors. Each value is multiplied by 16 prior to its load.
•@For example, 256 will be 4096.

gte_ldfc : Load a pointer representing a long vector to far color vector.•@

2.10. Load Instructions for Offset Value

gte_SetGeomOffset: Load offset value.

2.11. Load Instructions for Screen Location

gte_SetGeomScreen: Load screen location.

2.12. Load Rotation Matrix

gte_ldsvrtrow0 : Load SVECTOR to 1st row of the rotation matrix. Element m[1][0] of rotation matrix(row 2,column 1) is destroyed.

gte_SetRotMatrix : Load 3 X 3 matrix part of the structure MATRIX to rotation matrix.
•@

2.13. Load Instructions for Light Source Direction Matrix

gte_ldsvllrow0 : Load SVECTOR to the first row of light source direction matrix. Element m[1][0] of light source direction matrix (row 2,column 1) is destroyed.

gte_SetLightMatrix:Load 3 X 3 matrix part of the structure MATRIX to light source direction matrix.

2.14. Load Instructions for Light Source Color Matrix

gte_ldsvlcrow0 : Load SVECTOR to the first row of light source color matrix. Element m[1][0] of light source color matrix(row 2,column 1) is destroyed.

gte_SetColorMatrix: Load 3 X 3 matrix part of the structure MATRIX to light source color matrix.

2.15. Load Instructions for Translation Vector

gte_SetTransMatrix: Load vector part of structure MATRIX to translation vector.
gte_ldtr : Load long vector to translation vector.
gte_SetTransVector: Load long vector represented by pointer to translation vector.

2.16. Load Instructions for Interpolation Vector

gte_ld_intpol_uv0 :Load 1st vector for interpolation(2 dimensional byte vector) to far color vector.
Far color vector is destroyed.
gte_ld_intpol_uv1 :Load 2nd vector for interpolation (2 dimensional byte vector) to general short vector.
General short vector is destroyed.
gte_ld_intpol_bv0 :Same as gte_ld_intpol_uv0.
gte_ld_intpol_bv1 :Same as gte_ld_intpol_uv1.
gte_ld_intpol_sv0 :Load 1st vector for interpolation (3 dimensional byte vector) to far color vector.
Far color vector is destroyed.
gte_ld_intpol_sv1: Load 2nd vector for interpolation (3 dimensional byte vector) to general short vector.
General short vector is destroyed.

3. GTE Instructions

Instructions for GTE calculations

Abbreviations below will be used in the following description.
[ ] : a matrix or vector
[ ]*[ ] : a product of (matrix X matrix) or (matrix X vector).
dp[fc] : scalar multiplication
[rgb*sv] : termwise products

3.1. Instructions for Coordinates and Perspective Transformation

gte_rtps : pers(([rt]*[v0])>>12 + [tr]) -> sxy2

gte_rtpt : pers(([rt]*[v0])>>12 + [tr]) -> sxy0
pers(([rt]*[v1])>>12 + [tr]) -> sxy1
pers(([rt]*[v2])>>12 + [tr]) -> sxy2

3.2. Instructions for Matrix Operations

3.2.1. Instructions for Coordinate Conversion, Light Source Calculations

gte_rt : ([rt]*[v0])>>12 + [tr] -> lv
gte_ll : limit(([ll]*[v0])>>12) -> lv,sv
gte_lc : limit(([lc]*[sv])>>12) + [bk] -> lv,sv
gte_rtir_sf0 : [rt]*[sv] -> lv

3.2.2. Instructions for General Matrix Operations

The general matrix operations instructions are as follows;
Output Vector = Coefficient Matrix * Input Vector + Constant Vector

Coefficient matrix is (1,3,12), in other words, 12 bit fixed point numbers.
The 12 bit right shift follows (coefficient matrix * input vector).
Matrix or vector can take any of the following, and the result will be the same for all.

Coefficient Matrix : [rt] Rotation Matrix
: [ll] Light Source Direction Matrix
: [lc] Light Source Color Matrix
Input Vector : [v0] Vertex Coordinates Vector 0

```
                            : [v1]  Vertex Coordinates Vector 1
                            : [v2]  Vertex Coordinates Vector 2
                            : [sv]  General Short Vector
    Constant Vector                : [tr]  Translation Vector
                            : [bk]  Back Color Vector
                            : [fc]  Far Color Vector
    Output Vector           : [lv]  General Long Vector
                            : [sv]  General Short Vector

    gte_rtv0                       : ([rt]*[v0])>>12      -> lv,sv
    gte_rtv1                       : ([rt]*[v1])>>12      -> lv,sv
    gte_rtv2                       : ([rt]*[v2])>>12      -> lv,sv
    gte_rtir                       : ([rt]*[sv])>>12      -> lv,sv

    gte_rtv0tr              : ([rt]*[v0])>>12 + [tr] -> lv,sv
    gte_rtv1tr              : ([rt]*[v1])>>12 + [tr] -> lv,sv
    gte_rtv2tr              : ([rt]*[v2])>>12 + [tr] -> lv,sv
    gte_rtirtr              : ([rt]*[sv])>>12 + [tr] -> lv,sv

    gte_rtv0bk              : ([rt]*[v0])>>12 + [bk] -> lv,sv
    gte_rtv1bk              : ([rt]*[v1])>>12 + [bk] -> lv,sv
    gte_rtv2bk              : ([rt]*[v2])>>12 + [bk] -> lv,sv
    gte_rtirbk              : ([rt]*[sv])>>12 + [bk] -> lv,sv

    gte_rtv0fc              : ([rt]*[v0])>>12 + [fc] -> lv,sv
    gte_rtv1fc              : ([rt]*[v1])>>12 + [fc] -> lv,sv
    gte_rtv2fc              : ([rt]*[v2])>>12 + [fc] -> lv,sv
    gte_rtirfc              : ([rt]*[sv])>>12 + [fc] -> lv,sv

    gte_llv0                       : ([ll]*[v0])>>12 ->  lv,sv
    gte_llv1                       : ([ll]*[v1])>>12 ->  lv,sv
    gte_llv2                       : ([ll]*[v2])>>12 ->  lv,sv
    gte_llir                       : ([ll]*[sv])>>12 ->  lv,sv

    gte_llv0tr              : ([ll]*[v0])>>12 + [tr] -> lv,sv
    gte_llv1tr              : ([ll]*[v1])>>12 + [tr] -> lv,sv
    gte_llv2tr              : ([ll]*[v2])>>12 + [tr] -> lv,sv
    gte_llirtr              : ([ll]*[sv])>>12 + [tr] -> lv,sv

    gte_llv0bk              : ([ll]*[v0])>>12 + [bk] -> lv,sv
    gte_llv1bk              : ([ll]*[v1])>>12 + [bk] -> lv,sv
    gte_llv2bk              : ([ll]*[v2])>>12 + [bk] -> lv,sv
    gte_llirbk              : ([ll]*[sv])>>12 + [bk] -> lv,sv

    gte_llv0fc              : ([ll]*[v0])>>12 + [fc] -> lv,sv
    gte_llv1fc              : ([ll]*[v0])>>12 + [fc] -> lv,sv
    gte_llv2fc              : ([ll]*[v0])>>12 + [fc] -> lv,sv
    gte_llirfc              : ([ll]*[sv])>>12 + [fc] -> lv,sv

    gte_lcv0                       : ([lc]*[v0])>>12 ->  lv,sv
    gte_lcv1                       : ([lc]*[v1])>>12 ->  lv,sv
    gte_lcv2                       : ([lc]*[v2])>>12 ->  lv,sv
    gte_lcir                       : ([lc]*[sv])>>12 ->  lv,sv

    gte_lcv0tr              : ([lc]*[v0])>>12 + [tr] -> lv,sv
    gte_lcv1tr              : ([lc]*[v1])>>12 + [tr] -> lv,sv
    gte_lcv2tr              : ([lc]*[v2])>>12 + [tr] -> lv,sv
    gte_lcirtr              : ([lc]*[sv])>>12 + [tr] -> lv,sv

    gte_lcv0bk              : ([lc]*[v0])>>12 + [bk] -> lv,sv
    gte_lcv1bk              : ([lc]*[v1])>>12 + [bk] -> lv,sv
    gte_lcv2bk              : ([lc]*[v2])>>12 + [bk] -> lv,sv
    gte_lcirbk              : ([lc]*[sv])>>12 + [bk] -> lv,sv

    gte_lcv0fc              : ([lc]*[v0])>>12 + [fc] -> lv,sv
    gte_lcv1fc              : ([lc]*[v1])>>12 + [fc] -> lv,sv
    gte_lcv2fc              : ([lc]*[v2])>>12 + [fc] -> lv,sv
    gte_lcirfc              : ([lc]*[sv])>>12 + [fc] -> lv,sv
```

3.3. Instructions for Depth Queuing

```
    gte_dpcl                : (1-dp)[rgb*sv] + dp[fc] -> rgb,lv,sv
    gte_dpcs                : (1-dp)[rgb] + dp[fc] -> rgb,lv,sv
    gte_dpct                : (1-dp)[rgb0] + dp[fc] -> rgb0,lv,sv
                     (1-dp)[rgb1] + dp[fc] -> rgb1,lv,sv
```

```
                    (1-dp)[rgb2] + dp[fc] -> rgb2,lv,sv
```

3.4. Instructions for Interpolation

```
gte_intpl      : (1-dp)[sv] + dp[fc] -> rgb2,lv,sv
```

3.5. Instructions for Termwise Vector Square

```
gte_sqr12      : ((sv.vx^2)>>12,( sv.vy^2)>>12,( sv.vz^2)>>12) ->  lv,sv
gte_sqr0               : (sv.vx^2,sv.vy^2,sv.vz^2) ->  lv,sv
```

3.6. Instructions for Light Source Calculations

```
gte_ncs        : limit(([ ll]*[v0])>>12) ->  sv
                 limit(([ lc]*[ sv])>>12) + [ bk] ->rgb2
gte_nct        : limit(([ ll]*[v0])>>12) ->  sv
                 limit(([ lc]*[ sv])>>12) + [ bk] ->rgb0
                 limit(([ ll]*[v1])>>12) ->  sv
                 limit (([lc]*[ sv])>>12) + [ bk] ->rgb1
                 limit(([ ll]*[v2])>>12) ->  sv
                 limit(([ lc]*[ sv])>>12) + [ bk] ->rgb2
gte_ncds               : limit(([ ll]*[v0])>>12) ->  sv
                 limit(([ lc]*[ sv])>>12) + [ bk] -> sv
                 (1-dp)[rgb*sv] + dp[fc] -> rgb2
gte_ncdt               : limit(([ ll]*[v0])>>12) ->  sv
                 limit(([ lc]*[ sv])>>12) + [ bk] -> sv
                 (1-dp)[rgb*sv] + dp[fc] -> rgb0
                 limit(([ ll]*[v1])>>12) ->  sv
                 limit(([ lc]*[ sv])>>12) + [ bk] -> sv
                 (1-dp)[rgb*sv] + dp[fc] -> rgb1
                 limit(([ ll]*[v1])>>12) ->  sv
                 limit(([ lc]*[ sv])>>12) + [ bk] -> sv
                 (1-dp)[rgb*sv] + dp[fc] -> rgb2
gte_nccs               : limit(([ ll]*[v0])>>12) ->  sv
                 limit(([ lc]*[ sv])>>12) + [ bk] -> sv
                 [rgb*sv] -> rgb2
gte_ncct               : limit(([ ll]*[v0])>>12) ->  sv
                  limit(([  lc]*[ sv])>>12) + [ bk] -> sv
                  [ rgb*sv] -> rgb0
                limit(([ ll]*[v1])>>12) ->  sv
                  limit(([  lc]*[ sv])>>12) + [ bk] -> sv
                  [ rgb*sv] -> rgb1
                limit(([ ll]*[v2])>>12) ->  sv
                  limit(([  lc]*[ sv])>>12) + [ bk] -> sv
                  [ rgb*sv] -> rgb2
gte_cdp        : limit(([ lc]*[ sv])>>12) + [ bk] -> sv
                 (1-dp)[rgb*sv] + dp[fc] -> rgb2
gte_cc         : limit(([ lc]*[ sv])>>12) + [ bk] -> sv
                 [rgb*sv] -> rgb2
```

3.7. Instructions for Normal Clipping

```
gte_nclip      : sx0*sy1+sx1*sy2+sx2*sy0-sx0*sy2-sx1*sy0-sx2*sy1
                 = | sx1-sx0, sy1-sy0 | -> opz
                   | sx2-sx0, sy2-sy0 |
```

3.8. Instructions for Z Average

```
gte_avsz3      : (sz0+sz1+sz2)/3/4 ->  otz
gte_avsz4      : (sz0+sz1+sz2+sz3)/4/4 ->  otz
```

3.9. Instructions for Outer Products

```
gte_op12               : OuterProduct12( vect1,vect2) ->  lv,sv
gte_op0        : OuterProduct0( vect1,vect2) ->  lv,sv
```

3.10.Instructions for General Interpolation

```
gte_gpf12      : (dp[sv])>>12 ->  lv,sv
```

gte_gpf0 : dp[sv] -> lv,sv
gte_gpl12 : [lv] + (dp[sv])>>12 -> lv,sv
gte_gpl0 : [lv] + dp[sv] -> lv,sv

4. Store Instructions

Instructions for storing a value from GTE

4.1.Store Instructions for Screen Coordinates, X Y

gte_stsxy :Store 1 screen coordinate.
gte_stsxy3 :Store 3 screen coordinates to non-continuous address.
gte_stsxy3c :Store 3 screen coordinates to continuous address.
gte_stsxy0 :Store screen coordinates 0.
gte_stsxy1 :Store screen coordinates 1.
gte_stsxy2 :Store screen coordinates 1.
gte_stsxy01 :Store screen coordinates 0 and 1 to non-continuous address.
gte_stsxy01c :Store screen coordinates 0 and 1 to continuous address.
gte_stsxy3_f3 :Store 3 screen coordinates to POLY_F3 screen coordinate area.
gte_stsxy3_g3 :Store 3 screen coordinates to POLY_G3 screen coordinate area.
gte_stsxy3_ft3 :Store 3 screen coordinates to POLY_FT3 screen coordinate area.
gte_stsxy3_gt3 :Store 3 screen coordinates to POLY_GT3 screen coordinate area.
gte_stsxy3_f4 :Store 3 screen coordinates to POLY_F4 screen coordinate area.
gte_stsxy3_g4 :Store 3 screen coordinates to POLY_G4 screen coordinate area.
gte_stsxy3_ft4 :Store 3 screen coordinates to POLY_FT4 screen coordinate area.
gte_stsxy3_gt4 :Store 3 screen coordinates to POLY_GT4 screen coordinate area.

4.2. Store Instructions for Depth Queuing Values

gte_stdp :Store depth queuing values.

4.3. Store Instructions for Flags

gte_stflg :Store flag.
gte_stflg_4 :Store flag &0x00040000.

4.4. Store Instructions for Screen Coordinate Z

gte_stsz :Store 1 screen coordinate.
gte_stsz3 :Store 3 screen coordinates to non-continuous address.
gte_stsz4 :Store 3 screen coordinates to non-continuous address.
gte_stsz3c :Store 3 screen coordinates to continuous address.
gte_stsz4c :Store 4 screen coordinates to non-continuous address.

4.5. Store Instructions for OTZ

gte_stszotz :Store screen coordinates Z/4.
(Use screen coordinates Z instead of OTZ)
gte_stotz :Store OTZ.

4.6. Store Instructions for OPZ

gte_stopz :Store OPZ.

4.7. Store Instructions for General Short Vector

gte_stlvl :Store general short vector to structure VECTOR.
gte_stsv :Store general short vector to structure SVECTOR.
gte_stclmv :Store general short vector to the first column of structure MATRIX.
gte_stbv :Store 1st and 2nd elements of general short vector to structure byte vector.
gte_stcv :Store 1st, 2nd, and 3rd elements of general short vector to

structure CVECTOR.

4.8. Store Instructions for General Long Vector

gte_stlvnl :Store general long vector to structure VECTOR.
gte_stlvnl0 :Store 1st element of general long vector to structure VECTOR.
gte_stlvnl1 :Store 2nd element of general long vector to structure VECTOR.
gte_stlvnl2 :Store 3rd element of general long vector to structure VECTOR.

4.9. Store Instructions for RGB FIFO

gte_strgb :Store 1st word of RGB FIFO.
gte_strgb3 :Store 1st, 2nd, and 3rd words of RGB FIFO to non-continuous address.
gte_strgb3_g3 :Store 1st, 2nd, and 3rd words of RGB FIFO to POLY_G3 RGB area.
gte_strgb3_gt3 :Store 1st, 2nd, and 3rd words of RGB FIFO to POLY_GT3 RGB area.
gte_strgb3_g4 :Store 1st, 2nd, and 3rd words of RGB FIFO to POLY_G4 RGB area.
gte_strgb3_gt4 :Store 1st, 2nd, and 3rd words of RGB FIFO to POLY_GT4 RGB area.

4.10. Store Instructions for Offset Value

gte_ReadGeomOffset :Store offset values.

4.11. Store Instructions for Screen Position

gte_ReadGeomScreen :Store screen position.

4.12. Store Instructions for Rotation Matrix

gte_ReadRotMatrix :Store rotation matrix and translation vector to structure •@ MATRIX.
gte_sttr :Store translation vector to structure VECTOR.

4.13. Store Instructions for Light Source Direction Matrix

gte_ReadLightMatrix:Store light source direction matrix and back color vector •@ to structure MATRIX.

4.14. Store Instructions for Light Source Color Matrix

gte_ReadColorMatrix :Store light source color matrix and back color vector •@ to structure MATRIX.
gte_stfc :Store far color vector to structure VECTOR.

4.15. Store Instructions for LZC Value

gte_stlzc :Store LZC value.

5. Move Instructions

Instructions to move a value from GTE to GTE.

5.1 Move Instructions from Long Vector to Translation Vector

gte_mvlvtr :Move long vector to translation vector.

6. Others

6.1. NOP

gte_nop :NOP

6.2. Vector Operation Instructions

gte_subdvl : DVECTOR - DVECTOR -> VECTOR
gte_subdvd : DVECTOR - DVECTOR -> DECTOR
gte_adddvl : DVECTOR + DVECTOR -> VECTOR
gte_adddvd : DVECTOR + DVECTOR -> DECTOR

6.3. Rotation Matrix Flip Instructions

gte_FlipRotMatrixX: Multiple X row of rotation matrix by -1 to change the sign.

6.4. Translation Vector Flip Instructions

gte_FlipTRX :Multiple X row of translation vector by -1 to change the sign.

GTE inline function reference manual

1. register load functions

1.1.
gte_ldv0 Load vertex or normal to vertex register 0.

Syntax gte_ldv0(v)
SVECTOR *v;

Explanation Load vertex or normal to vertex register 0.
’¸“_ƒxƒNƒgƒ‹,Ü,½,Í–@•üƒxƒNƒgƒ‹, ð’¸“_ƒŒƒWƒXƒ^,Ì,O,É
ƒ••[ƒh,·,é

1.2.
gte_ldv1 Load vertex or normal to vertex register 1.

Syntax gte_ldv1(v)
SVECTOR *v;

Explanation Load vertex or normal to vertex register 1.
’¸“_ƒxƒNƒgƒ‹,Ü,½,Í–@•üƒxƒNƒgƒ‹, ð’¸“_ƒŒƒWƒXƒ^,Ì,P,É
ƒ••[ƒh,·,é

1.3.
gte_ldv2 Load vertex or normal to vertex register 2.

Syntax gte_ldv2(v)
SVECTOR *v;

Explanation Load vertex or normal to vertex register 2.
’¸“_ƒxƒNƒgƒ‹,Ü,½,Í–@•üƒxƒNƒgƒ‹, ð’¸“_ƒŒƒWƒXƒ^,Ì,Q,É
ƒ••[ƒh,·,é

1.4.
gte_ldv3 Load vertex or normal to vertex register 0,1,2.

Syntax gte_ldv3(v0,v1,v2)
SVECTOR * v0,*v1,*v2;

Explanation Load vertex or normal to vertex register 0,1,2.
,R,Â,Ì’¸“_ƒxƒNƒgƒ‹,Ü,½,Í–@•üƒxƒNƒgƒ‹, ð’¸“_ƒŒƒWƒXƒ^,Ì
,O•A,P•A,Q,Éƒ••[ƒh,·,é

1.5.
gte_ldv3c Load continuous vertex or normal to vertex register 0,1,2.

Syntax gte_ldv3c(v)

SVECTOR v[3];

Explanation
Load continuous vertex or normal to vertex register 0,1,2.
˜A‘±,µ,½,R,Â, Ì’¸“_ƒxƒNƒgƒ‹,Ü,½,Í–@•üƒxƒNƒgƒ‹,ð
’¸“_ƒŒƒWƒXƒ^,Ì,O•A,P•A,Q,Éƒ••[ƒh,·,é

1.5.0.1
gte_ldv3c_vertc

Load continuous vertex or normal to vertex register 0,1,2.

Syntax
gte_ldv3c_vertc(v)
VERTC   *v;

Explanation
Load continuous vertex or normal to vertex register 0,1,2
from VERTC structure(libgs.h)
˜A‘±,µ,½ ,R,Â,ÌVERTC•\‘¢‘Ì(libgs.h),Ì’¸“_ƒxƒNƒgƒ‹,Ü,½,Í
–@•üƒxƒNƒgƒ‹, ð’¸“_ƒŒƒWƒXƒ^,Ì,O•A,P•A,Q,Éƒ••[ƒh,·,é

1.5.1
gte_ldv01

Load vertex or normal to vertex register 0,1.

Syntax
gte_ldv01(v0,v1)
SVECTOR * v0,*v1;

Explanation
Load vertex or normal to vertex register 0,1.
,Q,Â,Ì’¸“_ƒxƒNƒgƒ‹,Ü,½,Í–@•üƒxƒNƒgƒ‹, ð’¸“_ƒŒƒWƒXƒ^,Ì
,O•A,P,Éƒ••[ƒh,·,é

1.5.2
gte_ldv01c

Load continuous vertex or normal to vertex register 0,1.

Syntax
gte_ldv01c(v)
SVECTOR v[2];

Explanation
Load continuous vertex or nor mal to vertex register 0,1.
˜A‘±,µ,½,Q,Â, Ì’¸“_ƒxƒNƒgƒ‹,Ü,½,Í–@•üƒxƒNƒgƒ‹,ð
’¸“_ƒŒƒWƒXƒ^,Ì,O•A,P,Éƒ••[ƒh,·,é

1.6.
gte_ldrgb

Load color and code to color register.

Syntax
gte_ldrgb(v)
CVECTOR *v;

Explanation
Load color and code to color register.
,q,f,aƒJƒ‰•[,ÆƒR•[ƒh, ðƒJƒ‰•[ƒŒƒWƒXƒ^,Éƒ••[ƒh,·,é

1.7.
gte_ldrgb3

Load color and code color fifo 0,1,2.

Syntax
gte_ldrgb3(v0,v1,v2)
CVECTOR *v0,*v1,*v2;

Explanation
Load color and code color fifo 0,1,2.
,R,Â,Ì,q,f, aƒJƒ‰•[,ÆƒR•[ƒh, ðƒJƒ‰•[fifo,O•A,P•A,Q,É
ƒ••[ƒh,·,é

1.7.1.
gte_ldrgb3c

Load color and code color fifo 0,1,2 from continuous address.

Syntax
gte_ldrgb3c(v0,v1,v2)
CVECTOR *v0,*v1,*v2;

Explanation
Load color and code color fifo 0,1,2 from continuous address.
,R,Â,Ì,q,f, aƒJƒ‰•[,ÆƒR•[ƒh, ð˜A‘±,·,é,R,Â, ÌƒAƒhƒŒƒX,©,ç
ƒJƒ‰•[fifo,O•A,P•A,Q,Éƒ••[ƒh,·,é

1.8.
gte_ldlv0

Load LS16 bit of VECTOR to vertex register 0.

Syntax
gte_ldlv0(v)
VECTOR *v;

Explanation
Load LS16 bit of VECTOR to vertex register 0.
,R,QbitƒxƒNƒgƒ‹,Ì‰º^Ê16 bit, ð’¸“_ƒŒƒWƒXƒ^,Ì,O,Éƒ••[ƒh,·,é

1.9.
gte_ldlvl Load LS16 bit of VECTOR to 16 bit universal vector.

Syntax gte_ldlvl(v)
VECTOR *v;

Explanation Load LS16 bit of VECTOR to 16 bit universal vector.
,R,QbitƒxƒNƒgƒ‹,Ì‰º^Ê16 bit,ð16 bit”Ä—pƒxƒNƒgƒ‹,É
ƒ••[ƒh,·,é

1.10.
gte_ldsv Load SVECTOR to 16 bit universal vector.

Syntax gte_ldsv(v)
SVECTOR *v;

Explanation Load SVECTOR to 16 bit universal vector.
16 bitƒxƒNƒgƒ‹,ð16 bit”Ä—pƒxƒNƒgƒ‹,Éƒ••[ƒh,·,é

1.11.
gte_ldbv Load byte vector to 16 bit universal vector.

Syntax gte_ldbv(v)
char v[2];

Explanation Load byte vector to 16 bit universal vector.
ByteƒxƒNƒgƒ‹,ð16 bit”Ä—pƒxƒNƒgƒ‹,Éƒ••[ƒh,·,é

1.12.
gte_ldcv Load CVECTOR to 16 bit universal vector.

Syntax gte_ldcv(v)
CVECTOR *v;

Explanation Load CVECTOR to 16 bit universal vector.
ƒJƒ‰•[ƒxƒNƒgƒ‹,ð16 bit”Ä—pƒxƒNƒgƒ‹,Éƒ••[ƒh,·,é

1.13.
gte_ldclmv Load column vector of MATRIX to universal register.

Syntax gte_ldclmv(m)
MATRIX *m;

Explanation Load column vector of MATRIX to universal register.
ƒ}ƒgƒŠƒNƒX, Ì•sƒxƒNƒgƒ‹,ð16 bit”Ä—pƒŒƒWƒXƒ^,Éƒ••[ƒh,·,é

1.14.
gte_lddp Load depth queuing value.

Syntax gte_lddp(p)
long p;

Explanation Load depth queuing value, p.
ƒfƒvƒXƒLƒ…•[ƒCƒ“ƒO—p,•’l,ðƒ••[ƒh,·,é

1.15.
gte_ldsxy3 Load screen XY-coordinates.

Syntax gte_ldsxy3(sxy0,sxy1,sxy2)
long sxy0,sxy1,sxy2;

Explanation Load screen XY-coordinates.
ƒXƒNƒŠ•[ƒ“,w, x•À•W,ðƒ••[ƒh,·,é

1.15.1.
gte_ldsxy3c Load screen XY-coord inates from continuos address.

Syntax gte_ldsxy3c(sxy0)
long *sxy0;

Explanation Load screen XY-coordinates from continuos address.
ƒXƒNƒŠ•[ƒ“,w, x•À•W,ð˜A‘±,µ,½ƒ•ƒ,ƒŠ— Ì^æ,©,çƒ••[ƒh,·,é

1.15.2.
gte_ldsxy0 Load screen XY-coordinate 0.

Syntax gte_ldsxy0(sxy)
long *sxy;

Explanation Load screen XY-coordinate 0.
ƒXƒNƒŠ•[ƒ“,w, x•À•W,O,ðƒ••[ƒh,·,é

1.15.3.
gte_ldsxy1 Load screen XY-coordinate 1.

Syntax gte_ldsxy1(sxy)
long *sxy;

Explanation Load screen XY-coordinate 1.
ƒXƒNƒŠ• [ƒ“,w,x•À•W,P,ðƒ••[ƒh,·,é

1.15.4.
gte_ldsxy2 Load screen XY-coordinate 2.

Syntax gte_ldsxy2(sxy)
long *sxy;

Explanation Load screen XY-coordinate 2.
ƒXƒNƒŠ•[ƒ“,w, x•À•W,Q,ðƒ••[ƒh,·,é

1.16.
gte_ldsz3 Load screen Z-coordinates.

Syntax gte_ldsz3(sz0,sz1,sz2)
long sz0,sz1,sz2;ƒxƒNƒgƒ‹

Explanation Load screen Z-coordinates.
ƒXƒNƒŠ•[ƒ“, y•À•W,ðƒ••[ƒh,·,é

1.17.
gte_ldsz4 Load screen Z-coordinates.

Syntax gte_ldsz4(sz0,sz1,sz2,sz3)
long sz0,sz1,sz2,sz3;

Explanation Load screen Z-coordinates.
ƒXƒNƒŠ•[ƒ“, y•À•W,ðƒ••[ƒh,·,é

1.18.
gte_ldopv1 Load outer product 1st vector.

Syntax gte_ldopv1(v)
VECTOR *v;

Explanation Load outer product 1st vector.
ŠO•Ï,Ì‘æ,PƒxƒNƒgƒ‹,ðƒ••[ƒh,·,é
!!destroy Rotation Matrix in GTE.
!!GTE,Ì’è•”‰ñ“]ƒ} ƒgƒŠƒNƒX,ð”j‰ó,·,é

1.19.
gte_ldopv2 Load outer product 2nd vector.

Syntax gte_ldopv2(v)
VECTOR *v;

Explanation Load outer product 2nd vector.
ŠO•Ï,Ì‘æ,QƒxƒNƒgƒ‹,ðƒ••[ƒh,·,é

1.20.
gte_ldlzc Load 32bit LZC data.

Syntax gte_ldlzc(data)
long data;

Explanation Load 32bit LZC data.
LZCƒŒƒWƒXƒ^,É,R, Qbitƒf•[ƒ^,ðƒ••[ƒh,·,é

1.21.
gte_SetRGBcd Load color and code to color register.

Syntax gte_SetRGBcd(v)
CVECTOR *v;

Explanation Load color and code to color register.
,q,f,aƒJƒ‰•[,ÆƒR•[ƒh, ðƒJƒ‰•[ƒŒƒWƒXƒ^,Éƒ••[ƒh,·,é

1.21.1.
gte_ldbkdir Load back color.

Syntax gte_ldbkdir(r,g,b)
long r,g,b;

Explanation Load back color.
ƒoƒbƒNƒJƒ‰•[,ð,»,Ì,Ü,Üƒ••[ƒh,·,é

1.22.
gte_SetBackColor Load back color x1 6.

Syntax gte_SetBackColor(r,g,b)
long r,g,b;

Explanation Load back color multiplied by 16 (x16).
(To match with the GTE operation format)
ƒoƒbƒNƒJƒ‰•[,ð,P,U”{,µ,Äƒ••[ƒh,·,é
(,f,s,d,Ì‰‰ŽZƒtƒH•[ƒ} ƒbƒg,É•‡,í,¹,é,½,ß)

1.22.1.
gte_ldfcdir Load far color.

Syntax gte_ldfcdir(r,g,b)
long r,g,b;

Explanation Load far color.
ƒtƒ@•[ƒJƒ‰•[,ð,»,Ì,Ü,Üƒ••[ƒh,·,é

1.23.
gte_SetFarColor Load far color.

Syntax gte_SetFarColor(r,g,b)
long r,g,b;

Explanation Load far color multiplied by 16 (x16).
(To match with the GTE operation format)
ƒtƒ@•[ƒJƒ‰•[,ð,P,U”{,µ,Äƒ••[ƒh,·,é
(,f,s,d,Ì‰‰ŽZƒtƒH•[ƒ} ƒbƒg,É•‡,í,¹,é,½,ß)

1.24.
gte_SetGeomOffset Load GTE-offset.

Syntax gte_SetGeomOffset(ofx,ofy)
long ofx,ofy;

Explanation Load GTE-offset.
GTEƒIƒtƒZƒbƒg,ðƒ••[ƒh,·,é

1.25.
gte_SetGeomScreen Load distance from viewpoint to screen.

Syntax gte_SetGeomScreen(h)
long h;

Explanation Load distance from viewpoint to screen.
Ž‹“_,©,çƒXƒNƒŠ•[ƒ“,Ü,Å,Ì‹——£,ðƒ••[ƒh,·,é

1.25.1.
gte_ldsvrtrow0 Load SVECTOR to row 0 of Rot Matrix

Syntax gte_ldsvrtrow0(v)
SVECTOR *v;

Explanation Load SVECTOR to row 0 of Rot Matrix
’è•”‰ñ“]ƒ} ƒgƒŠƒNƒX,Ì, P•s–Ú,ÉSVECTOR,ðƒ••[ƒh,·,é

1.26.
gte_SetRotMatrix Load Rotation Matrix.

Syntax gte_SetRotMatrix(m)
MATRIX *m;

Explanation Load Rotation Matrix.
’è•”‰ñ“]ƒ} ƒgƒŠƒNƒX,ðƒ••[ƒh,·,é

1.26.1.
gte_ldsvllrow0 Load SVECTOR to row 0 of Light Matrix

Syntax gte_ldsvllrow0(v)
SVECTOR *v;

Explanation Load SVECTOR to row 0 of Lig ht Matrix
’è•”ƒ‰ƒCƒgƒ} ƒgƒŠƒNƒX,Ì, P•s–Ú,ÉSVECTOR,ðƒ••[ƒh,·,é

1.27.
gte_SetLightMatrix Load Light Matrix.

Syntax gte_SetLightMatrix(m)
MATRIX *m;

Explanation Load Light Matrix.
’è•”ƒ‰ƒCƒgƒ} ƒgƒŠƒNƒX,ðƒ••[ƒh,·,é

1.27.1.
gte_ldsvlcrow0 Load SVECTOR to row 0 of Color Matrix

Syntax gte_ldsvlcrow0(v)
SVECTOR *v;

Explanation Load SVECTOR to row 0 of Color Matrix
’è•”ƒJƒ‰•[ƒ} ƒgƒŠƒNƒX,Ì, P•s–Ú,ÉSVECTOR,ðƒ••[ƒh,·,é

1.28.
gte_SetColorMatrix Load Color Matrix.

Syntax gte_SetColorMatrix(m)
MATRIX *m;

Explanation Load Color Matrix.
’è•”ƒJƒ‰•[ƒ} ƒgƒŠƒNƒX,ðƒ••[ƒh,·,é

1.28.1.
gte_ldtr Load Transfer vector by value

Syntax gte_ldtr(x,y,z)
long   x,y,z;

Explanation Load Transfer vector by value
’è•”•½•sˆÚ“®ƒxƒNƒgƒ‹, ð’l,Åƒ••[ƒh,·,é

1.29.
gte_SetTransMatrix Load Transfer vector.

Syntax gte_SetTransMatrix(m)
MATRIX *m;

Explanation Load Transfer vector.
’è•”•½•sˆÚ“®ƒxƒNƒgƒ‹,ðƒ••[ƒh,·,é

1.29.1
gte_SetTransVector Load Transfer vector.

Syntax gte_SetTransVector(v)
VECTOR *v;

Explanation Load Transfer vector.
’è•”•½•sˆÚ“®ƒxƒNƒgƒ‹,ðƒ••[ƒh,·,é

1.30.
gte_ld_intpol_uv0 Load byte vector to far color register for interpolation.

Syntax gte_ld_intpol_uv0(v)
char   v[2];

Explanation Load byte vector to far color register for interpolation.
“à‘}•ˆ—•,Ì,½,ß, ÉByteƒxƒNƒgƒ‹, ðƒtƒ@•[ƒJƒ‰•[ƒŒƒWƒXƒ^,É
ƒ••[ƒh,·,é

1.30.1
gte_ld_intpol_bv0 Load byte vector to far color register for interpolation.

Syntax gte_ld_intpol_bv0(v)
char v[2];

Explanation Load byte vector to far color register for interpolation.
“à‘}•ˆ—•,Ì,½,ß, ÉByteƒxƒNƒgƒ‹, ðƒtƒ@•[ƒJƒ‰•[ƒŒƒWƒXƒ^,É
ƒ••[ƒh,·,é

1.31.
gte_ld_intpol_uv1 Load byte vector to universal register for interpolation.

Syntax gte_ld_intpol_uv1(v)
char v[2];

Explanation Load byte vector to universal register for interpolation.
“à‘}•ˆ—•,Ì,½,ß, ÉByteƒxƒNƒgƒ‹,ð16 bit”Ä—pƒŒƒWƒXƒ^,É
ƒ••[ƒh,·,é

1.31.1
gte_ld_intpol_bv1 Load byte vector to universal register for interpolation.

Syntax gte_ld_intpol_bv1(v)
char v[2];

Explanation Load byte vector to universal register for interpolation.
“à‘}•ˆ—•,Ì,½,ß, ÉByteƒxƒNƒgƒ‹,ð16 bit”Ä—pƒŒƒWƒXƒ^,É
ƒ••[ƒh,·,é

1.32.
gte_ld_intpol_sv0 Load vertex to far color register for interpolation.

Syntax gte_ld_intpol_sv0(v)
SVECTOR v;

Explanation Load vertex to far color register for interpolation.
“à‘}•ˆ—•,Ì,½,ß, É’¸“_ƒxƒNƒgƒ‹, ðƒtƒ@•[ƒJƒ‰•[ƒŒƒWƒXƒ^,É
ƒ••[ƒh,·,é

1.33.
gte_ld_intpol_sv1 Load vertex to universal register for interpolation.

Syntax gte_ld_intpol_sv1(v)
SVECTOR v;

Explanation Load vertex to universal register for interpolation.
“à‘}•ˆ—•,Ì,½,ß, É’¸“_ƒxƒNƒgƒ‹,ð16 bit”Ä—pƒŒƒWƒXƒ^,É
ƒ••[ƒh,·,é

1.34.
gte_ldfc Load far color.

Syntax gte_ldfc(vc)
long vc[3];

Explanation Load far color.
ƒtƒ@•[ƒJƒ‰•[,ðƒ••[ƒh,·,é

2. GTE command

2.1.
gte_rtps kernel of RotTransPers

Syntax gte_rtps()

Explanation kernel of RotTransPers

2.2.
gte_rtpt kernel of RotTransPers3

Syntax	gte_rtpt()
Explanation	kernel of RotTransPers3

2.3.
gte_rt kernel of RotTrans

Syntax	gte_rt()
Explanation	kernel of RotTrans (Transfer vector)+(Rotation Matrix)*(vertex register 0)

2.4.
gte_rtv0 variation of gte_rt

Syntax	gte_rtv0()
Explanation	variation of gte_rt (Rotation Matrix)*(vertex register 0)

2.5.
gte_rtv1 variation of gte_rt

Syntax	gte_rtv1()
Explanation	variation of gte_rt (Rotation Matrix)*(vertex register 1)

2.6.
gte_rtv2 variation of gte_rt

Syntax	gte_rtv2()
Explanation	variation of gte_rt (Rotation Matrix)*(vertex register 2)

2.7.
gte_rtir variation of gte_rt

Syntax	gte_rtir()
Explanation	variation of gte_rt (Rotation Matrix)*(16 bit universal vector)

2.7.1.
gte_rtir_sf0 variation of gte_rt

Syntax	gte_rtir_sf0()
Explanation	variation of gte_rt (Rotation Matrix)*(16 bit universal vector) shift 0

2.7.2.
gte_rtv0tr general purpose matrix calculation

Syntax	gte_rtv0tr()
Explanation	[rt]*[v0]+[tr]

2.7.3.
gte_rtv1tr general purpose matrix calculation

Syntax	gte_rtv1tr()
Explanation	[rt]*[v1]+[tr]

2.7.4.
gte_rtv2tr general purpose matrix calculation

Syntax	gte_rtv2tr()
Explanation	[rt]*[v2]+[tr]

2.7.5.
gte_rtirtr general purpose matrix calculation

Syntax gte_rtirtr()

Explanation [rt]*[sv]+[tr]

2.7.6.
gte_rtv0bk general purpose matrix calculation

Syntax gte_rtv0bk()

Explanation [rt]*[v0]+[bk]

2.7.7.
gte_rtv1bk general purpose matrix calculation

Syntax gte_rtv1bk()

Explanation [rt]*[v1]+[bk]

2.7.8.
gte_rtv2bk general purpose matrix calculation

Syntax gte_rtv2bk()

Explanation [rt]*[v2]+[bk]

2.7.9.
gte_rtirbk general purpose matrix calculation

Syntax gte_rtirbk()

Explanation [rt]*[sv]+[bk]

2.7.10.
gte_rtv0fc general purpose matrix calculation

Syntax gte_rtv0fc()

Explanation [rt]*[v0]+[fc]

2.7.11.
gte_rtv1fc general purpose matrix calculation

Syntax gte_rtv1fc()

Explanation [rt]*[v1]+[fc]

2.7.12.
gte_rtv2fc general purpose matrix calculation

Syntax gte_rtv2fc()

Explanation [rt]*[v2]+[fc]

2.7.13.
gte_rtirfc general purpose matrix calculation

Syntax gte_rtirfc()

Explanation [rt]*[sv]+[fc]

2.8.
gte_ll kernel of LocalLight

Syntax gte_ll

Explanation kernel of LocalLight

2.8.1.
gte_llv0 general purpose matrix calculation

Syntax	gte_llv0()
Explanation	[ll]*[v0]

2.8.2.
gte_llv1 general purpose matrix calculation

Syntax	gte_llv1()
Explanation	[ll]*[v1]

2.8.3.
gte_llv2 general purpose matrix calculation

Syntax	gte_llv2()
Explanation	[ll]*[v2]

2.8.4.
gte_llir general purpose matrix calculation

Syntax	gte_llir()
Explanation	[ll]*[ir]

2.8.5.
gte_llv0tr general purpose matrix calculation

Syntax	gte_llv0tr()
Explanation	[ll]*[v0]+[tr]

2.8.6.
gte_llv1tr general purpose matrix calculation

Syntax	gte_llv1tr()
Explanation	[ll]*[v1]+[tr]

2.8.7.
gte_llv2tr general purpose matrix calculation

Syntax	gte_llv2tr()
Explanation	[ll]*[v2]+[tr]

2.8.8.
gte_llirtr general purpose matrix calculation

Syntax	gte_llirtr()
Explanation	[ll]*[sv]+[tr]

2.8.9.
gte_llv0bk general purpose matrix calculation

Syntax	gte_llv0bk()
Explanation	[ll]*[v0]+[bk]

2.8.10.
gte_llv1bk general purpose matrix calculation

Syntax	gte_llv1bk()
Explanation	[ll]*[v1]+[bk]

2.8.11.
gte_llv2bk general purpose matrix calculation

Syntax	gte_llv2bk()
Explanation	[ll]*[v2]+[bk]

2.8.12.

gte_llirbk general purpose matrix calculation

Syntax gte_llirbk()

Explanation [ll]*[sv]+[bk]

2.8.13.
gte_llv0fc general purpose matrix calculation

Syntax gte_llv0fc()

Explanation [ll]*[v0]+[fc]

2.8.14.
gte_llv1fc general purpose matrix calculation

Syntax gte_llv1fc()

Explanation [ll]*[v1]+[fc]

2.8.15.
gte_llv2fc general purpose matrix calculation

Syntax gte_llv2fc()

Explanation [ll]*[v2]+[fc]

2.8.16.
gte_llirfc general purpose matrix calculation

Syntax gte_llirfc()

Explanation [ll]*[sv]+[fc]

2.9.
gte_lc kernel of LightColor

Syntax gte_lc

Explanation kernel of LightColor

2.9.1.
gte_lcv0 general purpose matrix calculation

Syntax gte_lcv0()

Explanation [lc]*[v0]

2.9.2.
gte_lcv1 general purpose matrix calculation

Syntax gte_lcv1()

Explanation [lc]*[v1]

2.9.3.
gte_lcv2 general purpose matrix calculation

Syntax gte_lcv2()

Explanation [lc]*[v2]

2.9.4.
gte_lcir general purpose matrix calculation

Syntax gte_lcir()

Explanation [lc]*[sv]

2.9.5.
gte_lcv0tr general purpose matrix calculation

Syntax gte_lcv0tr()

Explanation [lc]*[v0]+[tr]

2.9.6.
gte_lcv1tr general purpose matrix calculation

Syntax gte_lcv1tr()

Explanation [lc]*[v1]+[tr]

2.9.7.
gte_lcv2tr general purpose matrix calculation

Syntax gte_lcv2tr()

Explanation [lc]*[v2]+[tr]

2.9.8.
gte_lcirtr general purpose matrix calculation

Syntax gte_lcirtr()

Explanation [lc]*[sv]+[tr]

2.9.9.
gte_lcv0bk general purpose matrix calculation

Syntax gte_lcv0bk()

Explanation [lc]*[v0]+[bk]

2.9.10.
gte_lcv1bk general purpose matrix calculation

Syntax gte_lcv1bk()

Explanation [lc]*[v1]+[bk]

2.9.11.
gte_lcv2bk general purpose matrix calculation

Syntax gte_lcv2bk()

Explanation [lc]*[v2]+[bk]

2.9.12.
gte_lcirbk general purpose matrix calculation

Syntax gte_lcirbk()

Explanation [lc]*[sv]+[bk]

2.9.13.
gte_lcv0fc general purpose matrix calculation

Syntax gte_lcv0fc()

Explanation [lc]*[v0]+[fc]

2.9.14.
gte_lcv1fc general purpose matrix calculation

Syntax gte_lcv1fc()

Explanation [lc]*[v1]+[fc]

2.9.15.
gte_lcv2fc general purpose matrix calculation

Syntax gte_lcv2fc()

Explanation [lc]*[v2]+[fc]

2.9.16.
gte_lcirfc general purpose matrix calculation

Syntax	gte_lcirfc()
Explanation	[lc]*[sv]+[fc]

2.10.
gte_dpcl kernel of DpqColorLight

Syntax	gte_dpcl()
Explanation	kernel of DpqColorLight

2.11.
gte_dpcs kernel of DpqColor

Syntax	gte_dpcs()
Explanation	kernel of DpqColor

2.12.
gte_dpct kernel of DpqColor3

Syntax	gte_dpct()
Explanation	kernel of DpqColor3

2.13.
gte_intpl kernel of Intpl

Syntax	gte_intpl()
Explanation	kernel of Intpl

2.14.
gte_sqr12 kernel of Square12

Syntax	gte_sqr12()
Explanation	kernel of Square12

2.15.
gte_sqr0 kernel of Square0

Syntax	gte_sqr0()
Explanation	kernel of Square0

2.16.
gte_ncs kernel of NormalColor

Syntax	gte_ncs()
Explanation	kernel of NormalColor

2.17.
gte_nct kernel of NormalColor3

Syntax	gte_nct()
Explanation	kernel of NormalColor3

2.18.
gte_ncds kernel of NormalColorDpq

Syntax	gte_ncds()
Explanation	kernel of NormalColorDpq

2.19.
gte_ncdt kernel of NormalColorDpq3

Syntax	gte_ncdt()
Explanation	kernel of NormalColorDpq3

2.20.

gte_nccs kernel of NormalColorCol

Syntax gte_nccs()

Explanation kernel of NormalColorCol

2.21.
gte_ncct kernel of NormalColorCol3

Syntax gte_ncct()

Explanation kernel of NormalColorCol3

2.22.
gte_cdp kernel of ColorDpq

Syntax gte_cdp()

Explanation kernel of ColorDpq

2.23.
gte_cc kernel of ColorCol

Syntax gte_cc()

Explanation kernel of ColorCol

2.24.
gte_nclip kernel of NormalClip

Syntax gte_nclip()

Explanation kernel of NormalClip

2.25.
gte_avsz3 kernel of AverageZ3

Syntax gte_avsz3()

Explanation kernel of AverageZ3

2.26.
gte_avsz4 kernel of AverageZ4

Syntax gte_avsz4()

Explanation kernel of AverageZ4

2.27.
gte_op12 kernel of OuterProduct12

Syntax gte_op12()

Explanation kernel of OuterProduct12

2.28.
gte_op0 kernel of OuterProduct0

Syntax gte_op0()

Explanation kernel of OuterProduct0

2.29.
gte_gpf12 first half of LoadAverage12

Syntax gte_gpf12()

Explanation first half of LoadAverage12

2.30.
gte_gpf0 first half of LoadAverage0

Syntax gte_gpf0()

Explanation first half of LoadAverage0

2.31.
gte_gpl12 last half of LoadAverage12

Syntax gte_gpl12()

Explanation last half of LoadAverage12

2.32.
gte_gpl0 last half of LoadAverage0

Syntax gte_gpl0()

Explanation last half of LoadAverage0

3. register store functions

3.1.
gte_stsxy Store screen xy.

Syntax gte_stsxy(sxy)
long *sxy;

Explanation Store sc reen xy.
ƒXƒNƒŠ•[ƒ“,w, x•À•W,ðƒXƒgƒA,·,é

3.1.1
gte_stsxy2 Store screen xy 2. = gte_stsxy(sxy)

Syntax gte_stsxy2(sxy)
long *sxy;

Explanation Store screen xy 2. = gte_stsxy(sxy)
ƒXƒNƒŠ•[ƒ“,w, x•À•W,Q,ðƒXƒgƒA,·,é = gte_stsxy(sxy)

3.1.2
gte_stsxy1 Store screen xy 1.

Syntax gte_stsxy1(sxy)
long *sxy;

Explanation Store screen xy 1.
ƒXƒNƒŠ•[ƒ“,w, x•À•W,P,ðƒXƒgƒA,·,é

3.1.3
gte_stsxy0 Store screen xy 0.

Syntax gte_stsxy0(sxy)
long *sxy;

Explanation Store screen xy 0.
ƒXƒNƒŠ• [ƒ“,w,x•À•W,O,ðƒXƒgƒA,·,é

3.2.
gte_stsxy3 Store screen xy 0,1,2.

Syntax gte_stsxy3(sxy0,sxy1,sxy2)
long *sxy0,*sxy1,*sxy2

Explanation Store screen xy 0,1,2.
ƒXƒNƒŠ•[ƒ“,w, x•À•W,O•A,P•A,Q, ðƒXƒgƒA,·,é

3.2.1.
gte_stsxy3_f3 Store screen xy 0,1,2. for POLY_F3

Syntax gte_stsxy3_f3(packet)
u_long *packet

Explanation Store screen xy 0,1,2. for POLY_F3
ƒXƒNƒŠ•[ƒ“,w, x•À•W,O•A,P•A,Q, ðPOLY_F3ƒpƒPƒbƒg,É
ƒXƒgƒA,·,é

3.2.2.
gte_stsxy3_g3 Store screen xy 0,1,2. for POLY_G3

Syntax	gte_stsxy3_g3(packet) u_long *packet
Explanation	Store screen xy 0,1,2. for POLY_G3 ƒXƒNƒŠ•[ƒ“,w, x•À•W,O•A,P•A,Q, ðPOLY_G3ƒpƒPƒbƒg,É ƒXƒgƒA,·,é

3.2.3.
gte_stsxy3_ft3 Store screen xy 0,1,2. for POLY_FT3

Syntax	gte_stsxy3_ft3(packet) u_long *packet
Explanation	Store screen xy 0,1,2. for POLY_FT3 ƒXƒNƒŠ•[ƒ“,w, x•À•W,O•A,P•A,Q, ðPOLY_FT3ƒpƒPƒbƒg,É ƒXƒgƒA,·,é

3.2.4.
gte_stsxy3_gt3 Store screen xy 0,1,2. for POLY_GT3

Syntax	gte_stsxy3_gt3(packet) u_long *packet
Explanation	Store screen xy 0,1,2. for POLY_GT3 ƒXƒNƒŠ•[ƒ“,w, x•À•W,O•A,P•A,Q, ðPOLY_GT3ƒpƒPƒbƒg,É ƒXƒgƒA,·,é

3.2.5.
gte_stsxy3_f4 Store screen xy 0,1,2. for POLY_F4

Syntax	gte_stsxy3_f4(packet) u_long *packet
Explanation	Store screen xy 0,1,2. for POLY_F4 ƒXƒNƒŠ•[ƒ“,w, x•À•W,O•A,P•A,Q, ðPOLY_F4ƒpƒPƒbƒg,É ƒXƒgƒA,·,é

3.2.6.
gte_stsxy3_g4 Store screen xy 0,1,2. for POLY_G4

Syntax	gte_stsxy3_g4(packet) u_long *packet
Explanation	Store screen xy 0,1,2. for POLY_G4 ƒXƒNƒŠ•[ƒ“,w, x•À•W,O•A,P•A,Q, ðPOLY_G4ƒpƒPƒbƒg,É ƒXƒgƒA,·,é

3.2.7.
gte_stsxy3_ft4 Store screen xy 0,1,2. for POLY_FT4

Syntax	gte_stsxy3_ft4(packet) u_long *packet
Explanation	Store screen xy 0,1,2. for POLY_FT4 ƒXƒNƒŠ•[ƒ“,w, x•À•W,O•A,P•A,Q, ðPOLY_FT4 ƒpƒPƒbƒg,ÉƒXƒgƒA,·,é

3.2.8.
gte_stsxy3_gt4 Store screen xy 0,1,2. for POLY_GT4

Syntax	gte_stsxy3_gt4(packet) u_long *packet
Explanation	Store screen xy 0,1,2. for POLY_GT4 ƒXƒNƒŠ•[ƒ“,w, x•À•W,O•A,P•A,Q, ðPOLY_GT4 ƒpƒPƒbƒg,ÉƒXƒgƒA,·,é

3.2.9.
gte_stsxy3c Store screen xy 0,1,2 to continuous 2D vertex.

Syntax	gte_stsxy3c(sxy) long sxy[3];

Explanation Store screen xy 0,1,2 to continuous 2D vertex.
ƒXƒNƒŠ•[ƒ“,w, x•À•W,O•A,P•A,Q, ð˜A‘±,µ,½,R,Â,Ì
,w,x•À•W,ÉƒXƒgƒA,·,é

3.2.10.
gte_stsxy01 Store screen xy 0,1.

Syntax gte_stsxy01(sxy0,sxy1)
long *sxy0,*sxy1

Explanation Store screen xy 0,1.
ƒXƒNƒŠ•[ƒ“,w, x•À•W,O•A,P, ðƒXƒgƒA,·,é

3.2.11.
gte_stsxy01c Store screen xy 0,1 to continuous 2D vertex.

Syntax gte_stsxy01c(sxy)
long sxy[2];

Explanation Store screen xy 0,1 to continuous 2D vertex.
ƒXƒNƒŠ•[ƒ“,w, x•À•W,O•A,P, ð˜A‘±,µ,½,Q,Â,Ì,w, x•À•W,É
ƒXƒgƒA,·,é

3.3.
gte_stdp Store depth queuing p.

Syntax gte_stdp(p)
long *p;

Explanation Store depth queuing p.
ƒfƒvƒXƒLƒ…•[ƒCƒ“ƒO—p,•’l, ðƒXƒgƒA,·,é

3.4.
gte_stflg Store flag.

Syntax gte_stflg(flag)
long *flag;

Explanation Store flag.
ƒtƒ‰ƒO,ðƒXƒgƒA,·,é

3.5.
gte_stsz Store screen z.

Syntax gte_stsz(sz)
long *sz;

Explanation Store screen z.
ƒXƒNƒŠ•[ƒ“, y•À•W,ðƒXƒgƒA,·,é

3.6.
gte_stsz3 Store screen z 0,1,2.

Syntax gte_stsz3(sz0,sz1,sz2)
long *sz0,*sz1,*sz2;

Explanation Store screen z 0,1,2.
ƒXƒNƒŠ•[ƒ“, y•À•W,O•A,P•A,Q, ðƒXƒgƒA,·,é

3.7.
gte_stsz4 Store screen z 0,1,2,3.

Syntax gte_stsz4(sz0,sz1,sz2,sz3)
long *sz0,*sz1,*sz2,*sz3;

Explanation Store screen z 0,1,2,3.
ƒXƒNƒŠ•[ƒ“, y•À•W,O•A,P•A,Q•A,R, ðƒXƒgƒA,·,é

3.7.1.
gte_stsz4c Store screen z 0,1,2,3. to continuos address

Syntax gte_stsz4c(sz0)
long *sz0;

Explanation

Store screen z 0,1,2,3. to continuos address
ƒXƒNƒŠ•[ƒ“, y•À•W,O•A,P•A,Q•A,R, ð˜A‘±,µ,½ƒ•ƒ,ƒŠ— Ìˆæ,É
ƒXƒgƒA,·,é

3.7.2.

gte_stsz3c Store screen z 0,1,2. to continuos address

Syntax

```
gte_stsz3c( sz0)
long    *sz0;
```

Explanation

Store screen z 0,1,2. to continuos address
ƒXƒNƒŠ•[ƒ“, y•À•W,O•A,P•A,Q, ð˜A‘±,µ,½ƒ•ƒ,ƒŠ— Ìˆæ,É
ƒXƒgƒA,·,é

3.8.

gte_stszotz Store screen z/4 as otz

Syntax

```
gte_stszotz( otz)
long    *otz;
```

Explanation

Store screen z/4 as otz
ƒXƒNƒŠ•[ƒ“, y•À•W/,S,ð,n,s,y, Ì‘ã,í,è,ÉƒXƒgƒA,·,é

3.9.

gte_stotz Store OTZ.

Syntax

```
gte_stotz( otz)
long    *otz;
```

Explanation

Store OTZ.
,n,s,y,ðƒXƒgƒA,·,é

3.10.

gte_stopz Store outer product.

Syntax

```
gte_stopz( opz)
long    *opz;
```

Explanation

Store outer product.
ŠO•Ï’l,n,o,y, ðƒXƒgƒA,·,é

3.11.

gte_stlvl Store VECTOR from 16 bit universal register.

Syntax

```
gte_stlvl(v)
VECTOR  *v;
```

Explanation

Store VECTOR from 16 bit universal register.
16 bit”Ä—pƒŒƒWƒXƒ^,©,ç,R, QbitƒxƒNƒgƒ‹, ðƒXƒgƒA,·,é

3.12.

gte_stlvnl Store VECTOR from 32bit universal register.

Syntax

```
gte_stlvnl(v)
VECTOR  *v;
```

Explanation

Store VECTOR from 32bit universal register.
,R,Qbit”Ä—pƒŒƒWƒXƒ^,©,ç,R, QbitƒxƒNƒgƒ‹, ðƒXƒgƒA,·,é

3.12.1.

gte_stlvnl0 Store from 32bit universal register.

Syntax

```
gte_stlvnl0(x)
long    *x;
```

Explanation

Store 1st component from 32bit universal register.
,R,Qbit”Ä—pƒŒƒWƒXƒ^,©, ç‘æ,P•¬•ª,ðƒXƒgƒA,·,é

3.12.2.

gte_stlvnl1 Store from 32bit universal register.

Syntax

```
gte_stlvnl1(x)
long    *x;
```

Explanation

Store 2nd component from 32bit universal register.

,R,Qbit”Ä–pƒŒƒWƒXƒ^,©, ç‘æ,Q•¬•ª,ðƒXƒgƒA,·,é

3.12.3.
gte_stlvnl2

Store from 32bit universal register.

Syntax

```
gte_stlvnl2(x)
long    *x;
```

Explanation

Store 2nd component from 32bit universal register.
,R,Qbit”Ä–pƒŒƒWƒXƒ^,©, ç‘æ,R•¬•ª,ðƒXƒgƒA,·,é

3.13.
gte_stsv

Store SVECTOR from 16 bit universal register.

Syntax

```
gte_stsv(v)
SVECTOR *v;
```

Explanation

Store SVECTOR from 16 bit universal register.
16 bit”Ä–pƒŒƒWƒXƒ^,©,ç16 bitƒxƒNƒgƒ‹, ðƒXƒgƒA,·,é

3.14.
gte_stclmv

Store MATRIX column from 16 bit universal register.

Syntax

```
gte_stclmv(m)
MATRIX  *m;
```

Explanation

Store MATRIX column from 16 bit universal register.
16 bit”Ä–pƒŒƒWƒXƒ^,©,ç16 bitƒxƒNƒgƒ‹,ðƒ} ƒgƒŠƒNƒX,Ì
•sƒxƒNƒgƒ‹, ÉƒXƒgƒA,·,é

3.15.
gte_stbv

Store Byte vector from LS8bit of 16 bit universal register.

Syntax

```
gte_stbv(v)
char    v[2];
```

Explanation

Store Byte vector from LS8bit of 16 bit universal register.
16 bit”Ä–pƒŒƒWƒXƒ^,Ì‰ºˆÊ, Wbit,ðƒoƒCƒgƒxƒNƒgƒ‹, ÉƒXƒgƒA,·,é

3.16.
gte_stcv

Store CVECTOR from LS8bit of 16 bit universal register.

Syntax

```
gte_stcv(v)
CVECTOR *v;
```

Explanation

Store CVECTOR from LS8bit of 16 bit universal register.
16 bit”Ä–pƒŒƒWƒXƒ^,Ì‰ºˆÊ, Wbit,ð,WbitƒxƒNƒgƒ‹, ÉƒXƒgƒA,·,é

3.17.
gte_strgb

Store CVECTOR from color register.

Syntax

```
gte_strgb(v)
CVECTOR *v;
```

Explanation

Store CVECTOR from color register.
ƒJƒ‰•[ƒŒƒWƒXƒ^,©,ç, WbitƒxƒNƒgƒ‹, ðƒXƒgƒA,·,é

3.18.
gte_strgb3

Store CVECTOR 0,1,2 from color   fifo.

Syntax

```
gte_strgb3( v0,v1,v2)
CVECTOR *v0,*v1,*v2;
```

Explanation

Store CVECTOR 0,1,2 from color   fifo.
ƒJƒ‰•[fifo,©,ç,WbitƒxƒNƒgƒ‹,O•A,P•A,Q, ðƒXƒgƒA,·,é

3.18.1.
gte_strgb3_g3

Store CVECTOR 0,1,2 from color   fifo to POLY_G3 packet.

Syntax

```
gte_strgb3_g3(packet)
u_long  *packet
```

Explanation

Store CVECTOR 0,1,2 from color   fifo to POLY_G3 packet.
ƒJƒ‰•[fifo,©,ç,WbitƒxƒNƒgƒ‹,O•A,P•A,Q, ðPOLY_G3
ƒpƒPƒbƒg,ÉƒXƒgƒA,·,é

3.18.2.
gte_strgb3_gt3 Store CVECTOR 0,1,2 from color fifo to POLY_GT3 packet.

Syntax gte_strgb3_gt3(packet)
u_long *packet

Explanation Store CVECTOR 0,1,2 from color fifo to POLY_GT3 packet.
ƒJƒ‰•[fifo,©,ç,WbitƒxƒNƒgƒ‹,O•A,P•A,Q, ðPOLY_GT3
ƒpƒPƒbƒg,ÉƒXƒgƒA,·,é

3.18.3.
gte_strgb3_g4 Store CVECTOR 0,1,2 from color fifo to POLY_G4 packet.

Syntax gte_strgb3_g4(packet)
u_long *packet

Explanation Store CVECTOR 0,1,2 from color fifo to POLY_G4 packet.
ƒJƒ‰•[fifo,©,ç,WbitƒxƒNƒgƒ‹,O•A,P•A,Q, ðPOLY_G4
ƒpƒPƒbƒg,ÉƒXƒgƒA,·,é

3.18.4.
gte_strgb3_gt4 Store CVECTOR 0,1,2 fr om color fifo to POLY_GT4 packet.

Syntax gte_strgb3_gt4(packet)
u_long *packet

Explanation Store CVECTOR 0,1,2 from color fifo to POLY_GT4 packet.
ƒJƒ‰•[fifo,©,ç,WbitƒxƒNƒgƒ‹,O•A,P•A,Q, ðPOLY_GT4
ƒpƒPƒbƒg,ÉƒXƒgƒA,·,é

3.19.
gte_ReadGeomOffset Store GTE-offset.

Syntax gte_ReadGeomOffset(ofx,ofy)
long *ofx,*ofy;

Explanation Store GTE-offset.
GTEƒIƒtƒZƒbƒg’l, ðƒXƒgƒA,·,é

3.20.
gte_ReadGeomScreen Store distance from viewpoint to screen.

Syntax gte_ReadGeomScreen(h)
long *h;

Explanation Store distance from viewpoint to screen.
Ž‹“_,©,çƒXƒNƒŠ•[ƒ“,Ü,Å,Ì‹——£, ðƒXƒgƒA,·,é

3.21.
gte_ReadRotMatrix Store Rotation Matrix.

Syntax gte_ReadRotMatrix(m)
MATRIX *m;

Explanation Store Rotation Matrix.
’è•”‰ñ“]ƒ} ƒgƒŠƒNƒX, ðƒXƒgƒA,·,é

3.21.1.
gte_sttr Store Transfer Vector

Syntax gte_sttr(v)
VECTOR *v;

Explanation Store Transfer Vector.
’è•”•½•sˆÚ“®ƒxƒNƒgƒ‹, ðƒXƒgƒA,·,é

3.22.
gte_ReadLightMatrix Store Light Matrix.

Syntax gte_ReadLightMatrix(m)
MATRIX *m;

Explanation Store Light Matrix.

’è•”ƒ‰ƒCƒgƒ} ƒgƒŠƒNƒX‚ðƒXƒgƒA‚·‚é

3.23.
gte_ReadColorMatrix Store Color Matrix.

Syntax
```
gte_ReadColorMatrix(m)
MATRIX  *m;
```

Explanation
Store Color Matrix.
’è•”ƒJƒ‰•[ƒ} ƒgƒŠƒNƒX‚ðƒXƒgƒA‚·‚é

3.24.
gte_stlzc Store LZC.

Syntax
```
gte_stlzc( lzc)
long    *lzc;
```

Explanation
Store LZC.
LZC•o—Í’l‚ðƒXƒgƒA‚·‚é

3.25.
gte_stfc Store far color.

Syntax
```
gte_stfc( vc)
long vc[3];
```

Explanation
Store far color.
ƒtƒ@•[ƒJƒ‰•[‚ðƒXƒgƒA‚·‚é

4. register move functions

4.1.
gte_mvlvtr move 32bit universal vector to Transfer vector

Syntax
```
gte_mvlvtr()
```

Explanation
move 32bit universal vector to Transfer vector
‚R‚Qbit”Ä—pƒxƒNƒgƒ‹‚ð•½•sˆÚ“®ƒxƒNƒgƒ‹‚Éƒ€•[ƒu‚·‚é

5. miscellaneous

5.1.
gte_nop No operation.

Syntax
```
gte_nop()
```

Explanation
No operation.
‚È‚É‚à‚µ‚È‚¢

5.2.
gte_subdvl v3= v1-v2

Syntax
```
gte_subdvl( v1,v2,v3)
DVECTOR * v1,*v2
VECTOR  *v3
```

Explanation
v3= v1-v2

5.3.
gte_subdvd v3= v1-v2

Syntax
```
gte_subdvd( v1,v2,v3)
DVECTOR *v1,*v2
DVECTOR *v3
```

Explanation
v3= v1-v2

5.4.
gte_adddvl v3= v1+v2

Syntax
```
gte_adddvl( v1,v2,v3)
```

DVECTOR * v1,*v2
VECTOR *v3

Explanation v3= v1+v2

5.5.
gte_adddvd v3= v1+v2

Syntax gte_adddvd(v1,v2,v3)
DVECTOR * v1,*v2
DVECTOR *v3

Explanation v3= v1+v2

5.6.
gte_FlipRotMatrixX flip X-row of rotate matrix. (R11,R12,R13) -> (-R11,-R12,-R13)

Syntax gte_FlipRotMatrixX()

Explanation flip X-row of rotate matrix.
(R11,R12,R13) -> (-R11,-R12,-R13)
‰ñ“]ƒ}ƒgƒŠƒNƒX,Ì X •s,ð”½“]
(R11,R12,R13) -> (-R11,-R12,-R13)

5.6.1
gte_FlipTRX flip X of transfer vector. TRX -> -TRX

Syntax gte_FlipTRX()

Explanation flip X of transfer vector. TRX -> -TRX
•½•sˆÚ“®ƒxƒNƒgƒ‹,Ì X ,ð”½“] TRX -> -TRX

GTE inline macro reference manual

1. Simple functions

1.1.
gte_RotTransPers

Syntax gte_RotTransPers(r1,r2,r3,r4,r5)

Explanation *r5 is return value of RotTransPers()
*r5,ÍRotTransPers(),Ì•Ô, è’l

1.2.
gte_RotTransPers3

Syntax gte_RotTransPers3(r1,r2,r3,r4,r5,r6,r7,r8,r9)

Explanation *r9 is return value of RotTransPers3()
*r9,ÍRotTransPers3(),Ì•Ô, è’l

1.3.
gte_RotTrans

Syntax gte_RotTrans(r1,r2,r3)

Explanation
1.4.
gte_LocalLight

Syntax gte_LocalLight(r1,r2)

Explanation
1.5.
gte_LightColor

Syntax gte_LightColor(r1,r2)

Explanation

1.6.
gte_DpqColorLight

Syntax `gte_DpqColorLight( r1,r2,r3,r4)`

Explanation

1.7.
gte_DpqColor

Syntax `gte_DpqColor( r1,r2,r3)`

Explanation

1.8.
gte_DpqColor3

Syntax `gte_DpqColor3( r1,r2,r3,r4,r5,r6,r7)`

Explanation

1.9.
gte_Intpl

Syntax `gte_Intpl( r1,r2,r3)`

Explanation

1.10.
gte_Square12

Syntax `gte_Square12( r1,r2)`

Explanation No return value
•Ô,è’l,Í,È,¢

1.11.
gte_Square0

Syntax `gte_Square0( r1,r2)`

Explanation No return value
•Ô,è’l,Í,È,¢

1.12.
gte_NormalColor

Syntax `gte_NormalColor( r1,r2)`

Explanation

1.13.
gte_NormalColor3

Syntax `gte_NormalColor3( r1,r2,r3,r4,r5,r6)`

Explanation

1.14.
gte_NormalColorDpq

Syntax `gte_NormalColorDpq( r1,r2,r3,r4)`

Explanation

1.15.
gte_NormalColorDpq3

Syntax `gte_NormalColorDpq3( r1,r2,r3,r4,r5,r6,r7,r8)`

Explanation

1.16.
gte_NormalColorCol

Syntax `gte_NormalColorCol( r1,r2,r3)`

Explanation

1.17.
gte_NormalColorCol3

Syntax gte_NormalColorCol3(r1,r2,r3,r4,r5,r6,r7)

Explanation

1.18.
gte_ColorDpq

Syntax gte_ColorDpq(r1,r2,r3,r4)

Explanation

1.19.
gte_ColorCol

Syntax gte_ColorCol(r1,r2,r3)

Explanation

1.20.
gte_NormalClip

Syntax gte_NormalClip(r1,r2,r3,r4)

Explanation *r4 is return value of NormalClip()
*r4,ÍNormalClip() ,Ì•Ô, è’l

1.21.
gte_AverageZ3

Syntax gte_AverageZ3(r1,r2,r3,r4)

Explanation *r4 is return value of AverageZ3()
*r4,ÍAverageZ3() ,Ì•Ô, è’l

1.22.
gte_AverageZ4

Syntax gte_AverageZ4(r1,r2,r3,r4,r5)

Explanation *r5 is return value of AverageZ4()
*r5,ÍAverageZ4() ,Ì•Ô, è’l

1.23.
gte_OuterProduct12

Syntax gte_OuterProduct12(r1,r2,r3)

Explanation !!destroy Rotation Matrix in GTE(different from OuterProduct12)
!!GTE,Ì’è•”‰ñ“]ƒ} ƒgƒŠƒNƒX,ð,±,í, ·•iOuterProduct12,Æˆ Ù,È, é•j

1.24.
gte_OuterProduct0

Syntax gte_OuterProduct0(r1,r2,r3)

Explanation !!destroy Rotation Matrix in GTE(different from OuterProduct0)
!!GTE,Ì’è•”‰ñ“]ƒ} ƒgƒŠƒNƒX,ð,±,í, ·•iOuterProduct0,Æˆ Ù,È, é•j

1.25.
gte_Lzc

Syntax gte_Lzc(r1,r2)

Explanation *r2 is return value of Lzc()
*r2 ,ÍLzc() ,Ì•Ô, è’l

2. Combined functions

gte_Lzc

Syntax gte_Lzc(r1,r2)

Explanation 4 vertices functions(RotTransPers4,..) can't be replaced by equivalent macros because they use OR of flags after rtpt & rtps. Please write directly in your program.

2.1.
gte_RotAverage3

Syntax	gte_RotAverage3(r1,r2,r3,r4,r5,r6,r7,r8,r9)
Explanation	*r9 is return value of RotAverage3() *r9 ,Í RotAverage3() ,Ì•Ô, è’l

2.2.
gte_RotNclip3

Syntax	gte_RotNclip3(r1,r2,r3,r4,r5,r6,r7,r8,r9,r10)
Explanation	*r10 is return value of RotNclip3() *r10 ,Í RotNclip3() ,Ì•Ô, è’l

2.3.
gte_RotAverageNclip3

Syntax	gte_RotAverageNclip3(r1,r2,r3,r4,r5,r6,r7,r8,r9,r10)
Explanation	*r10 is return value of RotAverageNclip3() *r10 ,Í RotAverageNclip3() ,Ì•Ô, è’l

2.4.
gte_RotColorDpq

Syntax	gte_RotColorDpq(r1,r2,r3,r4,r5,r6,r7)
Explanation	*r7 is return value of RotColorDpq() *r7 ,Í RotColorDpq() ,Ì•Ô, è’l

2.5.
gte_RotColorDpq3

Syntax
gte_RotColorDpq3(r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r15)

Explanation	*r15 is return value of RotColorDpq3() *r15 ,Í RotColorDpq3() ,Ì•Ô, è’l

2.6.
gte_RotAverageNclipColorDpq3

Syntax	gte_RotAverageNclipColorDpq3 (r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r15,r16)
Explanation	*r16 is return value of RotAverageNclipColorDpq3() *r16 ,Í RotAverageNclipColorDpq3() ,Ì•Ô, è’l

2.7.
gte_RotAverageNclipColorCol3

Syntax	gte_RotAverageNclipColorCol3 (r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r15,r16)
Explanation	*r16 is return value of RotAverageNclipColorCol3() *r16 ,Í RotAverageNclipColorCol3() ,Ì•Ô, è’l

2.8.
gte_LoadAverage12

Syntax	gte_LoadAverage12(r1,r2,r3,r4,r5)
Explanation	

2.9.
gte_LoadAverage0

Syntax	gte_LoadAverage0(r1,r2,r3,r4,r5)
Explanation	

2.10.

gte_LoadAverageShort12

Syntax gte_LoadAverageShort12(r1,r2,r3,r4,r5)

Explanation

2.11.
gte_LoadAverageShort0

Syntax gte_LoadAverageShort0(r1,r2,r3,r4,r5)

Explanation

2.12.
gte_LoadAverageByte

Syntax gte_LoadAverageByte(r1,r2,r3,r4,r5)

Explanation

2.13.
gte_LoadAverageCol

Syntax gte_LoadAverageCol(r1,r2,r3,r4,r5)

Explanation

3. Matrix functions

3.1.
gte_MulMatrix0

Syntax gte_MulMatrix0(r1,r2,r3)

Explanation !!destroy Rotation Matrix(same as MulMatrix0)
!!’è•”‰ñ“]ƒ} ƒgƒŠƒNƒX,ð,±,í,·(MulMatrix0,Æ“¯,¶•j

3.2.
gte_ApplyMatrix

Syntax gte_ApplyMatrix(r1,r2,r3)

Explanation !!destroy Rotation Matrix(same as ApplyMatrix)
!!’è•”‰ñ“]ƒ} ƒgƒŠƒNƒX,ð,±,í,·(ApplyMatrix,Æ“¯,¶•j

3.3.
gte_CompMatrix

Syntax gte_CompMatrix(r1,r2,r3)

Explanation !!destroy Rotation Matrix(same as CompMatrix)
!!destroy Transfer vector in GTE(different from CompMatrix)
!!’è•”‰ñ“]ƒ} ƒgƒŠƒNƒX,ð,±,í,·(CompMatrix,Æ“¯,¶•j
!!’è•”•½•s^Ú“®ƒxƒNƒgƒ‹,ð,±,í,·(CompMatrix,Æ“¯,¶•j

3.4.
gte_ApplyRotMatrix

Syntax gte_ApplyRotMatrix(r1,r2)

Explanation