

Inline Programming Reference

© 1998 Sony Computer Entertainment Inc.

Publication date: August 1998

Sony Computer Entertainment America
919 E. Hillsdale Blvd., 2nd floor
Foster City, CA 94404

Sony Computer Entertainment Europe
Waverley House
7-12 Noel Street
London W1V 4HH, England

The *Inline Programming Reference* manual is supplied pursuant to and subject to the terms of the Sony Computer Entertainment PlayStation® License and Development Tools Agreements, the Licensed Publisher Agreement and/or the Licensed Developer Agreement.

The *Inline Programming Reference* manual is intended for distribution to and use by only Sony Computer Entertainment licensed Developers and Publishers in accordance with the PlayStation® License and Development Tools Agreements, the Licensed Publisher Agreement and/or the Licensed Developer Agreement.

Unauthorized reproduction, distribution, lending, rental or disclosure to any third party, in whole or in part, of this book is expressly prohibited by law and by the terms of the Sony Computer Entertainment PlayStation® License and Development Tools Agreements, the Licensed Publisher Agreement and/or the Licensed Developer Agreement.

Ownership of the physical property of the book is retained by and reserved by Sony Computer Entertainment. Alteration to or deletion, in whole or in part, of the book, its presentation, or its contents is prohibited.

The information in the *Inline Programming Reference* manual is subject to change without notice. The content of this book is Confidential Information of Sony Computer Entertainment.

PlayStation and PlayStation logos are registered trademarks of Sony Computer Entertainment Inc. All other trademarks are property of their respective owners and/or their licensors.

Table of Contents

About This Manual	ix
Changes Since Last Release	ix
Related Documentation	ix
Manual Structure	ix
Developer Reference Series	x
Typographic Conventions	xi
Developer Support	xi
Chapter 1: GTE Inline Macro Library (DMPX)	
Overview	1-3
What is DMPX?	1-3
DMPX Release History	1-3
File Configuration	1-4
Syntax	1-4
Arguments	1-4
Options	1-4
Example	1-4
Overview	1-5
Inline Functions	1-5
Changing to Replaceable Functions	1-5
Inline Programming by Basic Functions	1-6
How to Use DMPX for Assembler Programs	1-7
How to set the environment	1-7
How to Install	1-7
How to Compile	1-7
With nop/Without nop Type 2 Commands	1-8
Include File Descriptions	1-8
inline_o.h	1-8
inline_c.h	1-8
inline_a.h	1-8
gtereg.h	1-9
gtemac.h	1-9
inline_a.h Header File for Assembler Programs	1-9
Compatibility of Old and New Type 2 Macros	1-10
DMPX Inline Function Table	1-11
Overview	1-11
GTE Register Descriptions	1-12
Load Instructions	1-13
GTE Instructions	1-15
Store Instructions	1-19
Move Instructions	1-21
Others	1-21
Chapter 2: GTE Inline Functions	
1. Register Load Functions	2-3
1.1. gte_ldv0	2-3
1.2. gte_ldv1	2-3
1.3. gte_ldv2	2-3
1.4. gte_ldv3	2-3
1.5. gte_ldv3c	2-3
1.5.0.1. gte_ldv3c_vrtc	2-3

1.5.1. gte_ldv01	2-4
1.5.2. gte_ldv01c	2-4
1.6. gte_ldrgb	2-4
1.7. gte_ldrgb3	2-4
1.7.1. gte_ldrgb3c	2-4
1.8. gte_ldlv0	2-4
1.9. gte_ldlv1	2-5
1.10. gte_ldsv	2-5
1.11. gte_ldbv	2-5
1.12. gte_ldcv	2-5
1.13. gte_ldclmv	2-5
1.14. gte_lddp	2-5
1.15. gte_ldsxy3	2-6
1.15.1. gte_ldsxy3c	2-6
1.15.2. gte_ldsxy0	2-6
1.15.3. gte_ldsxy1	2-6
1.15.4. gte_ldsxy2	2-6
1.16. gte_ldsz3	2-6
1.17. gte_ldsz4	2-7
1.18. gte_ldopv1	2-7
1.19. gte_ldopv2	2-7
1.20. gte_ldlzc	2-7
1.21. gte_SetRGBcd	2-7
1.21.1. gte_ldbkdir	2-7
1.22. gte_SetBackColor	2-8
1.22.1. gte_ldfcdir	2-8
1.23. gte_SetFarColor	2-8
1.24. gte_SetGeomOffset	2-8
1.25. gte_SetGeomScreen	2-8
1.25.1. gte_ldsvrtrow0	2-8
1.26. gte_SetRotMatrix	2-9
1.26.1. gte_ldsvllrow0	2-9
1.27. gte_SetLightMatrix	2-9
1.27.1. gte_ldsvlcrow0	2-9
1.28. gte_SetColorMatrix	2-9
1.28.1. gte_ldtr	2-9
1.29. gte_SetTransMatrix	2-10
1.29.1. gte_SetTransVector	2-10
1.30. gte_ld_intpol_uv0	2-10
1.30.1. gte_ld_intpol_bv0	2-10
1.31. gte_ld_intpol_uv1	2-10
1.31.1. gte_ld_intpol_bv1	2-10
1.32. gte_ld_intpol_sv0	2-11
1.33. gte_ld_intpol_sv1	2-11
1.34. gte_ldfc	2-11
2. GTE Commands	2-11
2.1. gte_rtps	2-11
2.2. gte_rtpt	2-11
2.3. gte_rt	2-11
2.4. gte_rtv0	2-12
2.5. gte_rtv1	2-12
2.6. gte_rtv2	2-12
2.7. gte_rtir	2-12
2.7.1. gte_rtir_sf0	2-12
2.7.2. gte_rtv0tr	2-12
2.7.3. gte_rtv1tr	2-13

2.7.4. gte_rtv2tr	2-13
2.7.5. gte_rtirtr	2-13
2.7.6. gte_rtv0bk	2-13
2.7.7. gte_rtv1bk	2-13
2.7.8. gte_rtv2bk	2-13
2.7.9. gte_rtirbk	2-14
2.7.10. gte_rtv0fc	2-14
2.7.11. gte_rtv1fc	2-14
2.7.12. gte_rtv2fc	2-14
2.7.13. gte_rtirfc	2-14
2.8. gte_ll	2-14
2.8.1. gte_llv0	2-15
2.8.2. gte_llv1	2-15
2.8.3. gte_llv2	2-15
2.8.4. gte_llir	2-15
2.8.5. gte_llv0tr	2-15
2.8.6. gte_llv1tr	2-15
2.8.7. gte_llv2tr	2-16
2.8.8. gte_llirtr	2-16
2.8.9. gte_llv0bk	2-16
2.8.10. gte_llv1bk	2-16
2.8.11. gte_llv2bk	2-16
2.8.12. gte_llirbk	2-16
2.8.13. gte_llv0fc	2-17
2.8.14. gte_llv1fc	2-17
2.8.15. gte_llv2fc	2-17
2.8.16. gte_llirfc	2-17
2.9. gte_lc	2-17
2.9.1. gte_lcv0	2-17
2.9.2. gte_lcv1	2-18
2.9.3. gte_lcv2	2-18
2.9.4. gte_lcir	2-18
2.9.5. gte_lcv0tr	2-18
2.9.6. gte_lcv1tr	2-18
2.9.7. gte_lcv2tr	2-18
2.9.8. gte_lcirtr	2-19
2.9.9. gte_lcv0bk	2-19
2.9.10. gte_lcv1bk	2-19
2.9.11. gte_lcv2bk	2-19
2.9.12. gte_lcirbk	2-19
2.9.13. gte_lcv0fc	2-19
2.9.14. gte_lcv1fc	2-20
2.9.15. gte_lcv2fc	2-20
2.9.16. gte_lcirfc	2-20
2.10. gte_dpcl	2-20
2.11. gte_dpcs	2-20
2.12. gte_dpct	2-20
2.13. gte_intpl	2-20
2.14. gte_sqr12	2-21
2.15. gte_sqr0	2-21
2.16. gte_ncs	2-21
2.17. gte_nct	2-21
2.18. gte_ncds	2-21
2.19. gte_ncdt	2-21
2.20. gte_nccs	2-22
2.21. gte_ncct	2-22

2.22. gte_cdp	2-22
2.23. gte_cc	2-22
2.24. gte_nclip	2-22
2.25. gte_avsz3	2-22
2.26. gte_avsz4	2-22
2.27. gte_op12	2-23
2.28. gte_op0	2-23
2.29. gte_gpf12	2-23
2.30. gte_gpf0	2-23
2.31. gte_gpl12	2-23
2.32. gte_gpl0	2-23
3. Register Store Functions	2-24
3.1. gte_stsxy	2-24
3.1.1. gte_stsxy2	2-24
3.1.2. gte_stsxy1	2-24
3.1.3. gte_stsxy0	2-24
3.2. gte_stsxy3	2-24
3.2.1. gte_stsxy3_f3	2-24
3.2.2. gte_stsxy3_g3	2-25
3.2.3. gte_stsxy3_ft3	2-25
3.2.4. gte_stsxy3_gt3	2-25
3.2.5. gte_stsxy3_f4	2-25
3.2.6. gte_stsxy3_g4	2-25
3.2.7. gte_stsxy3_ft4	2-25
3.2.8. gte_stsxy3_gt4	2-26
3.2.9. gte_stsxy3c	2-26
3.2.10. gte_stsxy01	2-26
3.2.11. gte_stsxy01c	2-26
3.3. gte_stdp	2-26
3.4. gte_stflg	2-26
3.5. gte_stsz	2-27
3.6. gte_stsz3	2-27
3.7. gte_stsz4	2-27
3.7.1. gte_stsz4c	2-27
3.7.2. gte_stsz3c	2-27
3.8. gte_stszotz	2-27
3.9. gte_stotz	2-28
3.10. gte_stopz	2-28
3.11. gte_stlvl	2-28
3.12. gte_stlvnl	2-28
3.12.1. gte_stlvnl0	2-28
3.12.2. gte_stlvnl1	2-28
3.12.3. gte_stlvnl2	2-29
3.13. gte_stsv	2-29
3.14. gte_stclmv	2-29
3.15. gte_stbv	2-29
3.16. gte_stcv	2-29
3.17. gte_strgb	2-29
3.18. gte_strgb3	2-30
3.18.1. gte_strgb3_g3	2-30
3.18.2. gte_strgb3_gt3	2-30
3.18.3. gte_strgb3_g4	2-30
3.18.4. gte_strgb3_gt4	2-30
3.19. gte_ReadGeomOffset	2-30
3.20. gte_ReadGeomScreen	2-31
3.21. gte_ReadRotMatrix	2-31

3.21.1. gte_sttr	2-31
3.22. gte_ReadLightMatrix	2-31
3.23. gte_ReadColorMatrix	2-31
3.24. gte_stlzc	2-31
3.25. gte_stfc	2-32
4. Register Move Functions	2-32
4.1. gte_mvlvtr	2-32
5. Miscellaneous	2-32
5.1. gte_nop	2-32
5.2. gte_subdvl	2-32
5.3. gte_subdvd	2-32
5.4. gte_adddvl	2-33
5.5. gte_adddvd	2-33
5.6. gte_FlipRotMatrixX	2-33
5.6.1. gte_FlipTRX	2-33

Chapter 3: GTE Inline Macros

1. Simple Functions	3-3
1.1. gte_RotTransPers	3-3
1.2. gte_RotTransPers3	3-3
1.3. gte_RotTrans	3-3
1.4. gte_LocalLight	3-3
1.5. gte_LightColor	3-3
1.6. gte_DpqColorLight	3-3
1.7. gte_DpqColor	3-3
1.8. gte_DpqColor3	3-4
1.9. gte_Intpl	3-4
1.10. gte_Square12	3-4
1.11. gte_Square0	3-4
1.12. gte_NormalColor	3-4
1.13. gte_NormalColor3	3-4
1.14. gte_NormalColorDpq	3-4
1.15. gte_NormalColorDpq3	3-5
1.16. gte_NormalColorCol	3-5
1.17. gte_NormalColorCol3	3-5
1.18. gte_ColorDpq	3-5
1.19. gte_ColorCol	3-5
1.20. gte_NormalClip	3-5
1.21. gte_AverageZ3	3-5
1.22. gte_AverageZ4	3-6
1.23. gte_OuterProduct12	3-6
1.24. gte_OuterProduct0	3-6
1.25. gte_Lzc	3-6
2. Combined Functions	3-6
2.1. gte_RotAverage3	3-6
2.2. gte_RotNclip3	3-6
2.3. gte_RotAverageNclip3	3-7
2.4. gte_RotColorDpq	3-7
2.5. gte_RotColorDpq3	3-7
2.6. gte_RotAverageNclipColorDpq3	3-7
2.7. gte_RotAverageNclipColorCol3	3-7
2.8. gte_LoadAverage12	3-7
2.9. gte_LoadAverage0	3-8
2.10. gte_LoadAverageShort12	3-8
2.11. gte_LoadAverageShort0	3-8
2.12. gte_LoadAverageByte	3-8

2.13. gte_LoadAverageCol	3-8
3. Matrix Functions	3-8
3.1. gte_MulMatrix0	3-8
3.2. gte_ApplyMatrix	3-8
3.3. gte_CompMatrix	3-9
3.4. gte_ApplyRotMatrix	3-9
Chapter 4: GTE Programming Guide	
Format	4-3
Registers	4-3
Register Access Instructions	4-3
Register Names	4-4
Commands	4-5
Delay Slots	4-5
Command Execution Cycles	4-5
Coding Limitations	4-5
Recommended Development Style	4-6
Chapter 5: GTE Register Specification	
Chapter 6: GTE Commands Reference	
Limiters	6-3
Calculation Error Detection	6-3
Explanations	6-4
Command List	6-5
Command Details	6-5

About This Manual

This manual is the first release of the *Inline Programming Reference* manual; it is current as of Run-Time Library 4.3. This manual brings together DMPSX, GTE inline macro and GTE register information that was previously available only in the following individual documents published on the Developer Support CD:

- Library Overview 3.7, Chapter 16: GTE Inline Macro Library
- Technical Note: GTE Inline Functions and DMPSX
- Technical Note: GTE Programming Guide
- Technical Note: GTE Register Specification
- Technical Note: GTE Command Reference

The purpose of this manual is to serve as a single reference for in-line programming using DMPSX, GTE inline macro and GTE register information for the PlayStation®.

Changes Since Last Release

Since the previous release of the DMPSX post processor tool (version 3.00) the following changes have been entered:

- The bug that caused a "hang-up" on the specific object has been fixed.
- The file specifications with relative path have been enabled.

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

Related Documentation

This manual should be read in conjunction with related documentation in the *Developer Reference Series*, including Run-Time Library release 4.3.

Note: The Developer Support Website posts current developments regarding the Libraries, and also provides notice of future documentation releases and upgrades.

Manual Structure

The *Inline Programming Reference* manual contains six chapters providing general descriptions of each of the high-level and low-level run-time libraries.

Section	Description
Ch. 1: DMPSX	Describes the DMPSX post processor tool and explains programming methods using the GTE inline functions and macros.
Ch. 2: GTE Inline Functions	Describes the inline functions that are provided for enhancing the speed of GTE processing.
Ch. 3: GTE Inline Macros	Describes the inline macros available for GTE processing.
Ch. 4: GTE Programming	Describes the programming procedures and techniques for programming the GTE directly, using DMPSX.
Ch. 5: GTE Register Specification	Describes the format and content of control and data registers of the GTE.
Ch. 6: GTE Command Reference	Describes the basic commands used by the GTE.

Developer Reference Series

This manual is part of the *Developer Reference Series*, a series of technical reference volumes covering all aspects of PlayStation development. The complete series is listed below:

Manual	Description
PlayStation Hardware	Describes the PlayStation hardware architecture and overviews its subsystems.
PlayStation Operating System	Describes the PlayStation operating system and related programming fundamentals.
Run-Time Library Overview	Describes the structure and purpose of the run-time libraries provided for PlayStation software development.
Run-Time Library Reference	Defines all available PlayStation run-time library functions, macros and structures.
Inline Programming Reference	Describes in-line programming using DMPSX, GTE inline macro and GTE register information.
SDevTC Development Environment	Describes the SDevTC (formerly "Psy-Q") Development Environment for PlayStation software development.
3D Graphics Tools	Describes how to use the PlayStation 3D Graphics Tools, including the animation and material editors.
Sprite Editor	Describes the Sprite Editor tool for creating sprite data and background picture components.
Sound Artist Tool	Provides installation and operation instructions for the DTL-H800 Sound Artist Board and explains how to use the Sound Artist Tool software.
File Formats	Describes all native PlayStation data formats.
Data Conversion Utilities	Describes all available PlayStation data conversion utilities, including both stand-alone and plug-in programs.
CD Emulator	Provides installation and operation instructions for the CD Emulator subsystem and related software.
CD-ROM Generator	Describes how to use the CD-ROM Generator software to write CD-R discs.
Performance Analyzer User Guide	Provides general instructions for using the Performance Analyzer software.
Performance Analyzer Technical Reference	Describes how to measure software performance and interpret the results using the Performance Analyzer.
DTL-H2000 Installation and Operation	Provides installation and operation instructions for the DTL-H2000 Development System.
DTL-H2500/2700 Installation and Operation	Provides installation and operation instructions for the DTL-H2500/H2700 Development Systems.

Typographic Conventions

Certain Typographic Conventions are used through out this manual to clarify the meaning of the text. The following conventions apply to all narrative text except for structure and function descriptions:

<i>Convention</i>	<i>Meaning</i>
<code>courier</code>	Indicates literal program code.
Bold	Indicates a document, chapter or section title.

The following conventions apply within structure and function descriptions only:

<i>Convention</i>	<i>Meaning</i>
Medium Bold	Denotes structure or function types and names.
<i>Italic</i>	Denotes function arguments and structure members.

Developer Support

Sony Computer Entertainment America (SCEA)

SCEA developer support is available to licensees in North America only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
<i>In North America</i> Attn: Developer Tools Coordinator Sony Computer Entertainment America 919 East Hillsdale Blvd., 2nd floor Foster City, CA 94404 Tel: (650) 655-8000	<i>In North America</i> E-mail: DevTech_Support@playstation.sony.com Web: http://www.scea.sony.com/dev Developer Support Hotline: (650) 655-8181 (Call Monday through Friday, 8 a.m. to 5 p.m., PST/PDT)

Sony Computer Entertainment Europe (SCEE)

SCEE developer support is available to licensees in Europe only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
<i>In Europe</i> Attn: Production Coordinator Sony Computer Entertainment Europe Waverley House 7-12 Noel Street London W1V 4HH Tel: +44 (0) 171 447 1600	<i>In Europe</i> E-mail: dev_support@playstation.co.uk Web: https://www-s.playstation.co.uk Developer Support Hotline: +44 (0) 171 447 1680 (Call Monday through Friday, 9 a.m. to 6 p.m., GMT or BST/BDT)

Chapter 1:

GTE Inline Macro Library (DMPSX)

Overview

The GTE Inline Macro Library offers the feature of controlling GTE without calling any functions. This chapter first describes the DMPSX program environment necessary for using the library and then explains the programming methods using the GTE Inline Macro Library.

What is DMPSX?

DMPSX is a post processor tool which helps improve the execution speed of PSX programs. The calling overhead can be eliminated by controlling GTE by including DMPSX unique header files into the programs, then using the inline functions (macros) defined in the include files.

The primary advantage of DMPSX is that library (libgte) function calls are eliminated from the programs, thereby reducing occurrences of Instruction Cache misses, and improving overall performance.

The compile procedures for programs using DMPSX differ from those of standard programs. A program is compiled to generate an object file **.obj*. This object file is input to *dmplx.exe*. Finally, the output of DMPSX is linked with other object files. Please refer to the later section *How to Compile* for details.

DMPSX Release History

Changes from Version 3.00

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

"dmplx.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 on the old versions of the DMPSX, and the old version of *inline.h* cannot be used for *DMPSX version 3.0*. To use a program created on an old version 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 previous 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'.

The following functions were added:

gte_ldrgb3c	gte_rtv0tr	gte_llv0	gte_lcv0	gte_stlvnl0
gte_ldbkdr	gte_rtv1tr	gte_llv1	gte_lcv1	gte_stlvnl1
gte_ldfcdtr	gte_rtv1tr	gte_llir	gte_lcv2	gte_stlvnl2
gte_ldsvrtrow0	gte_rtirtr	gte_llv0tr	gte_lcir	gte_mvlvtr
gte_ldsvllrow0	gte_rtv0bk	gte_llv1tr	gte_lcv0tr	
gte_ldsvlcrow0	gte_rtv1bk	gte_llv2tr	gte_lcv1tr	
gte_ldtr	gte_rtv2bk	gte_llirtr	gte_lcv2tr	
gte_rtirbk	gte_llv0bk	gte_lcirtr		
gte_rtv0fc	gte_llv1bk	gte_lcv0bk		
gte_rtv1fc	gte_llv2bk	gte_lcv1bk		
gte_rtv2fc	gte_llirbk	gte_lcv2bk		
gte_rtirfc	gte_llv0fc	gte_lcirbk		
	gte_llv1fc	gte_lcv0fc		
	gte_llv2fc	gte_lcv1fc		
	gte_llirfc	gte_lcv2fc		
		gte_lcirfc		

File Configuration

- **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**
Header file where 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

Arguments

DMPSX object-file [-o output-file] [-b]

Options

-o: Specify the output file name

Default: overwrite 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 source code with the GTE specified header files. Second, use the generated object file as the input of DMPSX. Then GTE code will be expanded in the object file.

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

(Note) Although the inline functions here have the same properties as the ones in the C language inline functions, they are generated differently.

Inline Functions

When a program is written using the libgte low level functions, such as RotTransPers, an instruction cache miss occurs, and thus the processing speed decreases. Using inline functions can avoid this situation. Execution time becomes faster. (Please note that when an original program runs on cache, using inline functions may delay its operation instead.)

GTE inline function names use the following naming convention.

gte_*****

GTE inline functions can be grouped roughly into two categories. One is basic functions for using GTE; the other is functions which can be replaced with the libgte low level functions just as they are. When the first letter following gte_ is a lower case letter, the function is a basic function. On the other hand, when it is a capital letter, the function may be replaced with a libgte low level function as it is.

gte_rtps	Basic function: invokes GTE commands, load/store gte registers
gte_RotTransPers	Replaceable function

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

Replaceable functions have a one-to-one correspondence with library functions.

Changing to Replaceable Functions

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

(1) Include the following header files

inline_o.h or inline_c.h
gtemac.h

(2) Check whether the function in question is a replaceable function by searching for the function name in the reference

(3) If the function name is in the reference,

1. Make the function an inline function by prefixing gte_ to the function name.

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

2. 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)

3. When a replaced inline function destroys GTE constants, such as Rotation Matrix or Transfer vector, 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.

(4) If not,

1. 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.

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

Within the reference:

Type 1 : Register Load Function is the loading of input data

Type 2 : GTE Command is execution

Type 3 : Register Store Function is the storage of output data

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

```

gte_ldv0(r1);           /*type1:load 3D coordinates*/
gte_rtps();             /*type2:Rotate,Transfer,Perspective*/
gte_stsxy(r2);          /*type3:store 2D coordinates*/
gte_stdp(r3);           /*type3:store depth queue 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 coordinates*/
gte_rtps();             /*type2:Rotate,Transfer,Perspective*/
gte_stsxy(r2);          /*type3:store 2D coordinates*/
gte_stszotz(r5);        /*type3:store sz/4 as otz*/

```

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

```

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

```

However, inserting too many CPU processes 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 of operation when a Type 1 function is called between Type 2 and Type 3 functions as shown below, so please do not do this.

Example of Incorrect Usage:

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

How to Use DMPSX for Assembler Programs

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

The following Type 2 instructions within inline_a.h require arguments.

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

Specify integers for all of these arguments. Please refer to the GTE documents for 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 DMPSX)

change makefile.mak 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
```

With nop/Without nop Type 2 Commands

The Type 2 commands (operations), which are macro defined in the header file, contain two command types within the same function: with nop and without nop. As a GTE specification, the cop2 command cannot be placed in the front two slots of the operation command. Therefore, standard Type 2 command macros have two nops. If a faster program becomes necessary, or you would like to delete nop, please use a macro without nop. In such cases, please make sure to program in such a way to allow the CPU commands to be entered in the front two slots of the Type 2 command.

The Type 2 command macros declared in every header file distinguish between the existence and non-existence of nop as follows:

(1) inline_c.h

_b is added to the end of a macro name for a macro without nop.

(e.g.)	with nop	gte_rtps()
	without nop	gte_rtps_b()

(2) inline_a.h

n_ is added to the beginning of the macro name for a macro with nop.

(e.g.)	with nop	nRTPS
	without nop	RTPS

(3) inline_o.h

Macros without nop are not used.

Include File Descriptions

inline_o.h

In the header file inline_o.h, Type 1 and 3 instructions of the old version inline.h header file are disclosed to retain 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 C program data to DMPSX, each inline includes one to four words of extra instructions.

inline_c.h

In this header file, the redundancies in 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 instructions (GET Register Load Instructions) and Type 3 instructions (GTE Register Store Instructions) are obsolete in the DMPSX from version 3.0. Thus non-DMPSX programs can also use these two instructions. The Type 2 instructions (GTE Commands) 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 to inline_o.h.

inline_a.h

This is the header file for Assembler programs. Only 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

This file defines GTE register macros 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 DMPSX. In this header file, the same macro function names with the same capabilities provided in the libgte.lib are defined as combinations of the basic DMPSX functions. Make use of this header file to start with, in making a libgte program high-speed using DMPSX.

inline_a.h Header File for Assembler Programs

inline_a.h is the header file which controls GTE in assembler programs. Only Type 2 commands are macro defined in inline_a.h. Type 1 and Type 3 commands are not defined.

Within the Type 2 commands macro defined in inline_a.h there are two types which are necessary per the GTE specifications: with nop and without nop. In GTE, cop2 commands cannot be placed in the front two slots of Type 2 commands. A name which begins with 'n' in the Type 2 commands supplied by inline_a.h has nop added to it before the command code in compliance with GTE specifications.

Command names which do not begin with an 'n' do not have nop attached. When such a command is used carelessly there is the possibility that code which can cause errors may be generated. Therefore, in dmplx.exe, when a cop2 command is confirmed in the first two slots of a command code, an error is output and processing terminates. When an error is output in dmplx, either change the command in question to a command with an 'n' attached or run a program which will allow at least two commands other than cop2 commands (CPU commands) to be input before the GTE code.

Also, some inline_a.h Type 2 commands use arguments. These commands use the arguments to control operation. Following is a list of the commands which use arguments. Refer to the GTE specifications for argument details and command activity.

MWMVA	sf, mx, v, cv, lm
SQR	sf
OP	sf
GPF	sf
GPL	sf

Compatibility of Old and New Type 2 Macros

In inline_a.h, the specifications for Type 2 commands between the previous inline_h and the currently released inline_c.h and inline_o.h C header files are different. Following is a list showing the compatibility of all Type 2 commands. Some assembler macros control activity by means of arguments. Refer to the GTE specifications for details on the arguments.

Table 1-1: Compatibility of Old and New Type 2 Macros

Macro for C	Macro for asm	Arguments (required items) sf, mx, v, cv, lm
gte_rtps()	RTPS	
gte_rtpt()	RTPT	
gte_rt()	MVMVA	1,0,0,0,0
gte_rtv0()	MVMVA	1,0,0,3,0
gte_rtv1()	MVMVA	1,0,1,3,0
gte_rtv2()	MVMVA	1,0,2,3,0
gte_rtir()	MVMVA	1,0,3,3,0
gte_rtir_sf0()	MVMVA	0,0,3,3,0
gte_rtv0tr()	MVMVA	1,0,0,0,0
gte_rtv1tr()	MVMVA	1,0,1,0,0
gte_rtv2tr()	MVMVA	1,0,2,0,0
gte_rtirtr()	MVMVA	1,0,3,0,0
gte_rtv0bk()	MVMVA	1,0,0,1,0
gte_rtv1bk()	MVMVA	1,0,1,1,0
gte_rtv2bk()	MVMVA	1,0,2,1,0
gte_rtirbk()	MVMVA	1,0,3,1,0
gte_ll()	MVMVA	1,1,0,3,1
gte_llv0()	MVMVA	1,1,0,3,0
gte_llv1()	MVMVA	1,1,1,3,0
gte_llv2()	MVMVA	1,1,2,3,0
gte_llir()	MVMVA	1,1,3,3,0
gte_llv0tr()	MVMVA	1,1,0,0,0
gte_llv1tr()	MVMVA	1,1,1,0,0
gte_llv2tr()	MVMVA	1,1,2,0,0
gte_llirtr()	MVMVA	1,1,3,0,0
gte_llv0bk()	MVMVA	1,1,0,1,0
gte_llv1bk()	MVMVA	1,1,1,1,0
gte_llv2bk()	MVMVA	1,1,2,1,0
gte_llirbk()	MVMVA	1,1,3,1,0
gte_lc()	MVMVA	1,2,3,1,1
gte_lcv0()	MVMVA	1,2,0,3,0
gte_lcv1()	MVMVA	1,2,1,3,0
gte_lcv2()	MVMVA	1,2,2,3,0
gte_lcir()	MVMVA	1,2,3,3,0
gte_lcv0tr()	MVMVA	1,2,0,0,0
gte_lcv1tr()	MVMVA	1,2,1,0,0
gte_lcv2tr()	MVMVA	1,2,2,0,0

gte_lcirtr()	MVMVA	1,2,3,0,0
gte_lcv0bk()	MVMVA	1,2,0,1,0
gte_lcv1bk()	MVMVA	1,2,1,1,0
gte_lcv2bk()	MVMVA	1,2,2,1,0
gte_lcirbk()	MVMVA	1,2,3,1,0
gte_dpcl()	DCPL	
gte_dpcs()	DPCS	
gte_dpct()	DPCT	
gte_intpl()	INTPL	
gte_sqr12()	SQR	1
gte_sqr0()	SQR	0
gte_ncs()	NCS	
gte_nct()	NCT	
gte_ncds()	NCDS	
gte_ncdt()	NCDT	
gte_nccs()	NCCS	
gte_ncct()	NCCT	
gte_cdp()	CDP	
gte_cc()	CC	
gte_nclip()	NCLIP	
gte_avsz3()	AVSZ3	
gte_avsz4()	AVSZ4	
gte_op12()	OP	1
gte_op0()	OP	0
gte_gpf12()	GPF	1
gte_gpf0()	GPF	0
gte_gpl12()	GPL	1
gte_gpl0()	GPL	0

DMPSX Inline Function Table

Overview

This table describes:

- GTE registers available for GTE
- How in-line functions can be sorted

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

GTE Register Descriptions

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

Table 1-2: GTE Register Symbols

Symbol	Contents
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 GPU codes
sv (= ir)	General 3 dimensional short vector
lv	General 3 dimensional long vector
dp	Short scalar for depth queuing (interpolation)
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

Load Instructions

Instruction for loading a value to GTE

Table 1-3: Load Instructions for Vertex Coordinates, Normal Line Vectors, etc.

Macro Name	Description
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.

Table 1-4: Load Instructions for RGB and GPU Code

Macro Name	Description
gte_ldrgb	Load CVECTOR to vector rgb. (GPU code is also overwritten)
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 rgb2 is loaded to rgb. (For GPU setting)
gte_SetRGBcd	=gte_ldrgb Load CVECTOR to vector rgb. (GPU code is also overwritten)

Table 1-5: Load Instructions for General Vectors

Macro Name	Description
gte_ldlv	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 CVECTOR to general short vector. GPU code parts will not be loaded anywhere.
gte_ldclmv	Load the first column of MATRIX to general short vector.

Table 1-6: Load Instructions for Depth Queuing Scalar Values

Macro Name	Description
gte_lddp	Load scalar values for depth queuing.

Table 1-7: Load Instructions for Screen Coordinates

Macro Name	Description
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.

<code>gte_ldsz3</code>	Load the screen coordinates Z of 3 vertices.
<code>gte_ldsz4</code>	Load the screen coordinates Z of 4 vertices.

Table 1-8: Load Instructions for Vector for Outer Products Calculation

Macro Name	Description
<code>gte_ldopv1</code>	Load 1st vector in order to calculate the outer products. (3 X 3 rotation matrix is destroyed.)
<code>gte_ldopv2</code>	Load 2nd vector in order to calculate the outer products. (General short vector is destroyed.)

Table 1-9: Load Instructions for LZC calculation

Macro Name	Description
<code>gte_ldlzc</code>	Load a value in order to calculate LZC (Leading Zero Counter). Numbers of 0 or 1 following the MSB of the value loaded are calculated.

Table 1-10: Load Instructions for Back Color Vector

Macro Name	Description
<code>gte_ldbkdir</code>	Load long vector value to back color vector.
<code>gte_SetBackColor</code>	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.

Table 1-11: Load Instructions for Far Color Vector

Macro Name	Description
<code>gte_ldfcdir</code>	Load long vector value to far color vector.
<code>gte_SetFarColor</code>	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.
<code>gte_ldfc</code>	Load a pointer representing a long vector to far color vector.

Table 1-12: Load Instructions for Offset Value

Macro Name	Description
<code>gte_SetGeomOffset</code>	Load offset value.

Table 1-13: Load Instructions for Screen Location

Macro Name	Description
<code>gte_SetGeomScreen</code>	Load screen location.

Table 1-14: Load Rotation Matrix

Macro Name	Description
<code>gte_ldsvrtrow0</code>	Load SVECTOR to 1st row of the rotation matrix. Element <code>m[1][0]</code> of rotation matrix (row 2,column 1) is destroyed.
<code>gte_SetRotMatrix</code>	Load 3 X 3 matrix part of the structure MATRIX to rotation matrix.

Table 1-15: Load Instructions for Light Source Direction Matrix

Macro Name	Description
<code>gte_ldsvllrow0</code>	Load SVECTOR to the first row of light source direction matrix. Element <code>m[1][0]</code> of light source direction matrix (row 2,column 1) is destroyed.
<code>gte_SetLightMatrix</code>	Load 3 X 3 matrix part of the structure MATRIX to light source direction matrix.

Table 1-16: Load Instructions for Light Source Color Matrix

Macro Name	Description
gte_Idsvlcrow0	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.

Table 1-17: Load Instructions for Translation Vector

Macro Name	Description
gte_SetTransMatrix	Load vector part of structure MATRIX to translation vector.
gte_Idtr	Load long vector to translation vector.
gte_SetTransVector	Load long vector represented by pointer to translation vector.

Table 1-18: Load Instructions for Interpolation Vector

Macro Name	Description
gte_Id_intpol_uv0	Load 1st vector for interpolation(2 dimensional byte vector) to far color vector. Far color vector is destroyed.
gte_Id_intpol_uv1	Load 2nd vector for interpolation (2 dimensional byte vector) to general short vector. General short vector is destroyed.
gte_Id_intpol_bv0	Same as gte_Id_intpol_uv0.
gte_Id_intpol_bv1	Same as gte_Id_intpol_uv1.
gte_Id_intpol_sv0	Load 1st vector for interpolation (3 dimensional byte vector) to far color vector. Far color vector is destroyed.
gte_Id_intpol_sv1	Load 2nd vector for interpolation (3 dimensional byte vector) to general short vector. General short vector is destroyed.

GTE Instructions

Instructions for GTE calculations

The 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

(1) Instructions for Coordinate and Perspective Transformation

Table 1-19: Instructions for Coordinate and Perspective Transformation

Macro Name	Description
gte_rtps	$\text{pers}([(rt]*[v0]) \gg 12 + [tr]) \rightarrow \text{sxy2}$
gte_rtp	$\text{pers}([(rt]*[v0]) \gg 12 + [tr]) \rightarrow \text{sxy0}$ $\text{pers}([(rt]*[v1]) \gg 12 + [tr]) \rightarrow \text{sxy1}$ $\text{pers}([(rt]*[v2]) \gg 12 + [tr]) \rightarrow \text{sxy2}$

(2) Instructions for Matrix Operations

Table 1-20: Instructions for Coordinate Conversion, Light Source Calculations

Macro Name	Description
gte_rt	$[(rt]*[v0]) \gg 12 + [tr] \rightarrow lv$
gte_ll	$\text{limit}([(ll]*[v0]) \gg 12) \rightarrow lv,sv$
gte_lc	$\text{limit}([(lc]*[sv]) \gg 12) + [bk] \rightarrow lv,sv$

```
gte_rtir_sf0      [rt]*[sv] -> lv
```

Instructions for General Matrix Operations

The general matrix operations instructions are as follows;

Output Vector = Coefficient Matrix * Input Vector + Constant Vector

Coefficient matrix elements are (1,3,12), in other words, 12 bit fixed point numbers. A 12 bit right shift follows (coefficient matrix * input vector).

Matrices or vectors can be 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
	: [lv] General Long Vector
	: [sv] General Short Vector

Table 1-21

Macro Name	Description
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

Table 1-22: Instructions for Depth Queuing

Macro Name	Description
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

Table 1-23: Instructions for Interpolation

Macro Name	Description
gte_intpl	(1-dp)[sv] + dp[fc] -> rgb2,lv,sv

Table 1-24: Instructions for Termwise Vector Square

Macro Name	Description
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

Table 1-25: Instructions for Light Source Calculations

Macro Name	Description
gte_ncs	limit(((ll)*[v0])>>12) -> sv
	limit(((ll)*[sv])>>12) + [bk] ->rgb2
gte_nct	limit(((ll)*[v0])>>12) -> sv
	limit(((ll)*[sv])>>12) + [bk] ->rgb0
	limit(((ll)*[v1])>>12) -> sv

	limit (([lc]*[sv])>>12) + [bk] ->rgb1
	limit(([l]*[v2])>>12) -> sv
	limit(([lc]*[sv])>>12) + [bk] ->rgb2
gte_ncds	limit(([l]*[v0])>>12) -> sv
	limit(([lc]*[sv])>>12) + [bk] -> sv
	(1-dp)[rgb*sv] + dp[fc] -> rgb2
gte_ncdt	limit(([l]*[v0])>>12) -> sv
	limit(([lc]*[sv])>>12) + [bk] -> sv
	(1-dp)[rgb*sv] + dp[fc] -> rgb0
	limit(([l]*[v1])>>12) -> sv
	limit(([lc]*[sv])>>12) + [bk] -> sv (1-dp)
	[rgb*sv] + dp[fc] -> rgb1
	limit(([l]*[v1])>>12) -> sv
	limit(([lc]*[sv])>>12) + [bk] -> sv
	(1-dp)[rgb*sv] + dp[fc] -> rgb2
gte_nccs	limit(([l]*[v0])>>12) -> sv
	limit(([lc]*[sv])>>12) + [bk] -> sv
	[rgb*sv] -> rgb2
gte_ncct	limit(([l]*[v0])>>12) -> sv
	limit(([lc]*[sv])>>12) + [bk] -> sv
	[rgb*sv] -> rgb0 limit(([l]*[v1])>>12) -> sv
	limit(([lc]*[sv])>>12) + [bk] -> sv
	[rgb*sv] -> rgb1
	limit(([l]*[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

Table 1-26: Instructions for Normal Clipping

Macro Name	Description
gte_nclip	$sx0*sy1+sx1*sy2+sx2*sy0-sx0*sy2-sx1*sy0-sx2*sy1$ $= sx1-sx0, sy1-sy0 \rightarrow opz$ $ sx2-sx0, sy2-sy0 $

Table 1-27: Instructions for Z Average

Macro Name	Description
gte_avsz3	$(sz0+sz1+sz2)/3/4 \rightarrow otz$
gte_avsz4	$(sz0+sz1+sz2+sz3)/4/4 \rightarrow otz$

Table 1-28: Instructions for Outer Products

Macro Name	Description
gte_op12	OuterProduct12(vect1,vect2) -> lv,sv
gte_op0	OuterProduct0(vect1,vect2) -> lv,sv

Table 1-29: Instructions for General Interpolation

Macro Name	Description
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

Store Instructions

Instructions for storing a value from GTE

Table 1-30: Store Instructions for Screen Coordinates X, Y

Macro Name	Description
gte_stsxy	Store 1 set of screen coordinates.
gte_stsxy3	Store 3 screen coordinates to non-continuous addresses.
gte_stsxy3c	Store 3 screen coordinates to continuous addresses.
gte_stsxy0	Store screen coordinates 0.
gte_stsxy1	Store screen coordinates 1.
gte_stsxy2	Store screen coordinates 2.
gte_stsxy01	Store screen coordinates 0 and 1 to non-continuous addresses.
gte_stsxy01c	Store screen coordinates 0 and 1 to continuous addresses.
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.

Table 1-31: Store Instructions for Depth Queuing Values

Macro Name	Description
gte_stdp	Store depth queuing values.

Table 1-32: Store Instructions for Flags

Macro Name	Description
gte_stflg	Store flag.
gte_stflg_4	Store flag &0x00040000.

Table 1-33: Store Instructions for Screen Coordinate Z

Macro Name	Description
gte_stsz	Store 1 set of screen coordinates.
gte_stsz3	Store 3 screen coordinates to non-continuous addresses.
gte_stsz4	Store 4 screen coordinates to non-continuous addresses.
gte_stsz3c	Store 3 screen coordinates to continuous addresses.
gte_stsz4c	Store 4 screen coordinates to continuous addresses.

Table 1-34: Store Instructions for OTZ

Macro Name	Description
gte_stszotz	Store screen coordinates Z/4. (Use screen coordinates Z instead of OTZ)
gte_stotz	Store OTZ.

Table 1-35: Store Instructions for OPZ

Macro Name	Description
gte_stopz	Store OPZ.

Table 1-36: Store Instructions for General Short Vector

Macro Name	Description
gte_stlvi	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.

Table 1-37: Store Instructions for General Long Vector

Macro Name	Description
gte_stlvni	Store general long vector to structure VECTOR.
gte_stlvni0	Store 1st element of general long vector to structure VECTOR.
gte_stlvni1	Store 2nd element of general long vector to structure VECTOR.
gte_stlvni2	Store 3rd element of general long vector to structure VECTOR.

Table 1-38: Store Instructions for RGB FIFO

Macro Name	Description
gte_strgb	Store 1st word of RGB FIFO.
gte_strgb3	Store 1st, 2nd, and 3rd words of RGB FIFO to non-continuous addresses.
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.

Table 1-39: Store Instructions for Offset Value

Macro Name	Description
gte_ReadGeomOffset	Store offset values.

Table 1-40: Store Instructions for Screen Position

Macro Name	Description
gte_ReadGeomScreen	Store screen position.

Table 1-41: Store Instructions for Rotation Matrix

Macro Name	Description
gte_ReadRotMatrix	Store rotation matrix and translation vector to structure MATRIX.
gte_sttr	Store translation vector to structure VECTOR.

Table 1-42: Store Instructions for Light Source Direction Matrix

Macro Name	Description
gte_ReadLightMatrix	Store light source direction matrix and back color vector to structure MATRIX.

Table 1-43: Store Instructions for Light Source Color Matrix

Macro Name	Description
gte_ReadColorMatrix	Store light source color matrix and back color vector to structure MATRIX.
gte_stfc	Store far color vector to structure VECTOR.

Table 1-44: Store Instructions for LZC Value

Macro Name	Description
gte_stlzc	Store LZC value.

Move Instructions

Instructions to move a value from GTE to GTE.

Table 1-45: Move Instructions from Long Vector to Translation Vector

Macro Name	Description
gte_mvlvtr	Move long vector to translation vector.

Others

Table 1-46: NOP

Macro Name	Description
gte_nop	NOP

Table 1-47: Vector Operation Instructions

Macro Name	Description
gte_subdvl	DVECTOR - DVECTOR -> VECTOR
gte_subdvd	DVECTOR - DVECTOR -> DVECTOR
gte_adddvl	DVECTOR + DVECTOR -> VECTOR
gte_adddvd	DVECTOR + DVECTOR -> DVECTOR

Table 1-48: Rotation Matrix Flip Instructions

Macro Name	Description
gte_FlipRotMatrixX	Multiply X row of rotation matrix by -1 to change the sign.

Table 1-49: Translation Vector Flip Instructions

Macro Name	Description
gte_FlipTRX	Multiply X component of translation vector by -1 to change the sign.

Chapter 2:

GTE Inline Functions

1. Register Load Functions

1.1. gte_ldv0

Syntax

```
gte_ldv0(v)  
SVECTOR *v;
```

Explanation

Load vertex or normal to vertex register 0.

1.2. gte_ldv1

Syntax

```
gte_ldv1(v)  
SVECTOR *v;
```

Explanation

Load vertex or normal to vertex register 1.

1.3. gte_ldv2

Syntax

```
gte_ldv2(v)  
SVECTOR *v;
```

Explanation

Load vertex or normal to vertex register 2.

1.4. gte_ldv3

Syntax

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

Explanation

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

1.5. gte_ldv3c

Syntax

```
gte_ldv3c(v)  
SVECTOR v[3];
```

Explanation

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

1.5.0.1. gte_ldv3c_vertc

Syntax

```
gte_ldv3c_vertc(v)  
VERTC *v;
```

Explanation

Load continuous vertex or normal to vertex register 0, 1, 2 from VERTC structure (libgs.h)

1.5.1. gte_ldv01**Syntax**

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

Explanation

Load vertex or normal to vertex register 0, 1.

1.5.2. gte_ldv01c**Syntax**

```
gte_ldv01c(v)
SVECTOR v[2];
```

Explanation

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

1.6. gte_ldrgb**Syntax**

```
gte_ldrgb(v)
CVECTOR *v;
```

Explanation

Load color and code to color register.

1.7. gte_ldrgb3**Syntax**

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

Explanation

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

1.7.1. gte_ldrgb3c**Syntax**

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

Explanation

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

1.8. gte_ldlv0**Syntax**

```
gte_ldlv0(v)
VECTOR *v;
```

Explanation

Load LS 16 bits of VECTOR to vertex register 0.

1.9. gte_ldlvl**Syntax**

```
gte_ldlvl(v)
VECTOR *v;
```

Explanation

Load LS 16 bits of VECTOR to 16 bit universal vector.

1.10. gte_ldsv**Syntax**

```
gte_ldsv(v)
SVECTOR *v;
```

Explanation

Load SVECTOR to 16 bit universal vector.

1.11. gte_ldbv**Syntax**

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

Explanation

Load byte vector to 16 bit universal vector.

1.12. gte_ldcv**Syntax**

```
gte_ldcv(v)
CVECTOR *v;
```

Explanation

Load CVECTOR to 16 bit universal vector.

1.13. gte_ldclmv**Syntax**

```
gte_ldclmv(m)
MATRIX *m;
```

Explanation

Load column vector of MATRIX to universal register.

1.14. gte_lddp**Syntax**

```
gte_lddp(p)
long p;
```

Explanation

Load depth queuing value, p.

1.15. gte_ldsxy3

Syntax

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

Explanation

Load screen XY-coordinates.

1.15.1. gte_ldsxy3c

Syntax

```
gte_ldsxy3c(sxy0)
long *sxy0;
```

Explanation

Load screen XY-coordinates from continuous addresses.

1.15.2. gte_ldsxy0

Syntax

```
gte_ldsxy0(sxy)
long *sxy;
```

Explanation

Load screen XY-coordinate 0.

1.15.3. gte_ldsxy1

Syntax

```
gte_ldsxy1(sxy)
long *sxy;
```

Explanation

Load screen XY-coordinate 1.

1.15.4. gte_ldsxy2

Syntax

```
gte_ldsxy2(sxy)
long *sxy;
```

Explanation

Load screen XY-coordinate 2.

1.16. gte_ldsz3

Syntax

```
gte_ldsz3(sz0,sz1,sz2)
long sz0, sz1, sz2;
```

Explanation

Load screen Z-coordinates.

1.17. gte_ldsz4**Syntax**

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

Explanation

Load screen Z-coordinates.

1.18. gte_ldopv1**Syntax**

```
gte_ldopv1(v)
VECTOR *v;
```

Explanation

Load outer product 1st vector.
Warning! Use of this function will destroy the Rotation Matrix in GTE.

1.19. gte_ldopv2**Syntax**

```
gte_ldopv2(v)
VECTOR *v;
```

Explanation

Load outer product 2nd vector.

1.20. gte_ldlzc**Syntax**

```
gte_ldlzc(data)
long data;
```

Explanation

Load 32-bit LZC data.

1.21. gte_SetRGBcd**Syntax**

```
gte_SetRGBcd(v)
CVECTOR *v;
```

Explanation

Load color and code to color register.

1.21.1. gte_ldbkdir**Syntax**

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

Explanation

Load back color.

1.22. gte_SetBackColor

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).

1.22.1. gte_ldfcdir

Syntax

gte_ldfcdir(r,g,b)

long r, g, b;

Explanation

Load far color.

1.23. gte_SetFarColor

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).

1.24. gte_SetGeomOffset

Syntax

gte_SetGeomOffset(ofx,ofy)

long ofx, ofy;

Explanation

Load GTE-offset.

1.25. gte_SetGeomScreen

Syntax

gte_SetGeomScreen(h)

long h;

Explanation

Load distance from viewpoint to screen.

1.25.1. gte_ldsvrtrow0

Syntax

gte_ldsvrtrow0(v)

SVECTOR *v;

Explanation

Load SVECTOR to row 0 of Rotation Matrix.

1.26. gte_SetRotMatrix**Syntax**

```
gte_SetRotMatrix(m)
MATRIX *m;
```

Explanation

Load Rotation Matrix.

1.26.1. gte_Idsvllrow0**Syntax**

```
gte_Idsvllrow0(v)
SVECTOR *v;
```

Explanation

Load SVECTOR to row 0 of Light Matrix.

1.27. gte_SetLightMatrix**Syntax**

```
gte_SetLightMatrix(m)
MATRIX *m;
```

Explanation

Load Light Matrix.

1.27.1. gte_Idsvlcrow0**Syntax**

```
gte_Idsvlcrow0(v)
SVECTOR *v;
```

Explanation

Load SVECTOR to row 0 of Color Matrix.

1.28. gte_SetColorMatrix**Syntax**

```
gte_SetColorMatrix(m)
MATRIX *m;
```

Explanation

Load Color Matrix.

1.28.1. gte_Idtr**Syntax**

```
gte_Idtr(x,y,z)
long x, y, z;
```

Explanation

Load Transfer Vector by value.

1.29. gte_SetTransMatrix

Syntax

```
gte_SetTransMatrix(m)  
MATRIX *m;
```

Explanation

Load Transfer vector.

1.29.1. gte_SetTransVector

Syntax

```
gte_SetTransVector(v)  
VECTOR *v;
```

Explanation

Load Transfer Vector.

1.30. gte_Id_intpol_uv0

Syntax

```
gte_Id_intpol_uv0(v)  
char v[2];
```

Explanation

Load byte vector to far color register for interpolation.

1.30.1. gte_Id_intpol_bv0

Syntax

```
gte_Id_intpol_bv0(v)  
char v[2];
```

Explanation

Load byte vector to far color register for interpolation.

1.31. gte_Id_intpol_uv1

Syntax

```
gte_Id_intpol_uv1(v)  
char v[2];
```

Explanation

Load byte vector to universal register for interpolation.

1.31.1. gte_Id_intpol_bv1

Syntax

```
gte_Id_intpol_bv1(v)  
char v[2];
```

Explanation

Load byte vector to universal register for interpolation.

1.32. gte_ld_intpol_sv0**Syntax**

```
gte_ld_intpol_sv0(v)
SVECTOR v;
```

Explanation

Load vertex to far color register for interpolation.

1.33. gte_ld_intpol_sv1**Syntax**

```
gte_ld_intpol_sv1(v)
SVECTOR v;
```

Explanation

Load vertex to universal register for interpolation.

1.34. gte_ldfc**Syntax**

```
gte_ldfc(vc)
long vc[3];
```

Explanation

Load far color.

2. GTE Commands**2.1. gte_rtps****Syntax**

```
gte_rtps()
```

Explanation

Kernel of RotTransPers.

2.2. gte_rtpt**Syntax**

```
gte_rtpt()
```

Explanation

Kernel of RotTransPers3.

2.3. gte_rt**Syntax**

```
gte_rt()
```

Explanation

Kernel of RotTrans
 $(\text{Transfer vector}) + (\text{Rotation Matrix}) * (\text{vertex register } 0).$

2.4. gte_rtv0

Syntax

`gte_rtv0()`

Explanation

Variation of `gte_rt`
 $(\text{Rotation Matrix}) * (\text{vertex register } 0).$

2.5. gte_rtv1

Syntax

`gte_rtv1()`

Explanation

Variation of `gte_rt`
 $(\text{Rotation Matrix}) * (\text{vertex register } 1).$

2.6. gte_rtv2

Syntax

`gte_rtv2()`

Explanation

Variation of `gte_rt`
 $(\text{Rotation Matrix}) * (\text{vertex register } 2).$

2.7. gte_rtir

Syntax

`gte_rtir()`

Explanation

Variation of `gte_rt`
 $(\text{Rotation Matrix}) * (16 \text{ bit universal vector}).$

2.7.1. gte_rtir_sf0

Syntax

`gte_rtir_sf0()`

Explanation

Variation of `gte_rt`
 $(\text{Rotation Matrix}) * (16 \text{ bit universal vector}) \text{ shift } 0.$

2.7.2. gte_rtv0tr

Syntax

`gte_rtv0tr()`

Explanation

General purpose matrix calculation
 $[rt] * [v0] + [tr]$.

2.7.3. gte_rtv1tr**Syntax**

`gte_rtv1tr()`

Explanation

General purpose matrix calculation
 $[rt] * [v1] + [tr]$.

2.7.4. gte_rtv2tr**Syntax**

`gte_rtv2tr()`

Explanation

General purpose matrix calculation
 $[rt] * [v2] + [tr]$.

2.7.5. gte_rtirtr**Syntax**

`gte_rtirtr()`

Explanation

General purpose matrix calculation
 $[rt] * [sv] + [tr]$.

2.7.6. gte_rtv0bk**Syntax**

`gte_rtv0bk()`

Explanation

General purpose matrix calculation
 $[rt] * [v0] + [bk]$.

2.7.7. gte_rtv1bk**Syntax**

`gte_rtv1bk()`

Explanation

General purpose matrix calculation
 $[rt] * [v1] + [bk]$.

2.7.8. gte_rtv2bk**Syntax**

`gte_rtv2bk()`

Explanation

General purpose matrix calculation
 $[rt][v2] + [bk]$.

2.7.9. gte_rtirbk

Syntax

`gte_rtirbk()`

Explanation

General purpose matrix calculation
 $[rt][sv] + [bk]$.

2.7.10. gte_rtv0fc

Syntax

`gte_rtv0fc()`

Explanation

General purpose matrix calculation
 $[rt][v0] + [fc]$.

2.7.11. gte_rtv1fc

Syntax

`gte_rtv1fc()`

Explanation

General purpose matrix calculation
 $[rt][v1] + [fc]$.

2.7.12. gte_rtv2fc

Syntax

`gte_rtv2fc()`

Explanation

General purpose matrix calculation
 $[rt][v2] + [fc]$.

2.7.13. gte_rtirfc

Syntax

`gte_rtirfc()`

Explanation

General purpose matrix calculation
 $[rt][sv] + [fc]$.

2.8. gte_ll

Syntax

`gte_ll`

Explanation

Kernel of LocalLight.

2.8.1. gte_llv0**Syntax**

gte_llv0()

Explanation

General purpose matrix calculation
 $[ll] * [v0]$.

2.8.2. gte_llv1**Syntax**

gte_llv1()

Explanation

General purpose matrix calculation
 $[ll] * [v1]$.

2.8.3. gte_llv2**Syntax**

gte_llv2()

Explanation

General purpose matrix calculation
 $[ll] * [v2]$.

2.8.4. gte_llir**Syntax**

gte_llir()

Explanation

General purpose matrix calculation
 $[ll] * [ir]$.

2.8.5. gte_llv0tr**Syntax**

gte_llv0tr()

Explanation

General purpose matrix calculation
 $[ll] * [v0] + [tr]$.

2.8.6. gte_llv1tr**Syntax**

gte_llv1tr()

Explanation

General purpose matrix calculation

$[ll] * [v1] + [tr]$.

2.8.7. gte_llv2tr

Syntax

gte_llv2tr()

Explanation

General purpose matrix calculation
 $[ll] * [v2] + [tr]$.

2.8.8. gte_llirtr

Syntax

gte_llirtr()

Explanation

General purpose matrix calculation
 $[ll] * [sv] + [tr]$.

2.8.9. gte_llv0bk

Syntax

gte_llv0bk()

Explanation

General purpose matrix calculation
 $[ll] * [v0] + [bk]$.

2.8.10. gte_llv1bk

Syntax

gte_llv1bk()

Explanation

General purpose matrix calculation
 $[ll] * [v1] + [bk]$.

2.8.11. gte_llv2bk

Syntax

gte_llv2bk()

Explanation

General purpose matrix calculation
 $[ll] * [v2] + [bk]$.

2.8.12. gte_llirbk

Syntax

gte_llirbk()

Explanation

General purpose matrix calculation
 $[ll] * [sv] + [bk]$.

2.8.13. gte_llv0fc**Syntax**

`gte_llv0fc()`

Explanation

General purpose matrix calculation
 $[ll] * [v0] + [fc]$.

2.8.14. gte_llv1fc**Syntax**

`gte_llv1fc()`

Explanation

General purpose matrix calculation
 $[ll] * [v1] + [fc]$.

2.8.15. gte_llv2fc**Syntax**

`gte_llv2fc()`

Explanation

General purpose matrix calculation
 $[ll] * [v2] + [fc]$.

2.8.16. gte_llirfc**Syntax**

`gte_llirfc()`

Explanation

General purpose matrix calculation
 $[ll] * [sv] + [fc]$.

2.9. gte_lc**Syntax**

`gte_lc`

Explanation

Kernel of LightColor.

2.9.1. gte_lcv0**Syntax**

`gte_lcv0()`

Explanation

General purpose matrix calculation
 $[lc] * [v0]$.

2.9.2. gte_lcv1

Syntax

`gte_lcv1()`

Explanation

General purpose matrix calculation
 $[lc]*[v1]$.

2.9.3. gte_lcv2

Syntax

`gte_lcv2()`

Explanation

General purpose matrix calculation
 $[lc]*[v2]$.

2.9.4. gte_lcir

Syntax

`gte_lcir()`

Explanation

General purpose matrix calculation
 $[lc]*[sv]$.

2.9.5. gte_lcv0tr

Syntax

`gte_lcv0tr()`

Explanation

General purpose matrix calculation
 $[lc]*[v0]+[tr]$.

2.9.6. gte_lcv1tr

Syntax

`gte_lcv1tr()`

Explanation

General purpose matrix calculation
 $[lc]*[v1]+[tr]$.

2.9.7. gte_lcv2tr

Syntax

`gte_lcv2tr()`

Explanation

General purpose matrix calculation
 $[lc]*[v2]+[tr]$.

2.9.8. gte_lcirtr**Syntax**

gte_lcirtr()

Explanation

General purpose matrix calculation
 $[lc] * [sv] + [tr]$.

2.9.9. gte_lcv0bk**Syntax**

gte_lcv0bk()

Explanation

General purpose matrix calculation
 $[lc] * [v0] + [bk]$.

2.9.10. gte_lcv1bk**Syntax**

gte_lcv1bk()

Explanation

General purpose matrix calculation
 $[lc] * [v1] + [bk]$.

2.9.11. gte_lcv2bk**Syntax**

gte_lcv2bk()

Explanation

General purpose matrix calculation
 $[lc] * [v2] + [bk]$.

2.9.12. gte_lcirbk**Syntax**

gte_lcirbk()

Explanation

General purpose matrix calculation
 $[lc] * [sv] + [bk]$.

2.9.13. gte_lcv0fc**Syntax**

gte_lcv0fc()

Explanation

General purpose matrix calculation
 $[lc] * [v0] + [fc]$.

2.9.14. gte_lcv1fc

Syntax

gte_lcv1fc()

Explanation

General purpose matrix calculation
 $[lc] * [v1] + [fc]$.

2.9.15. gte_lcv2fc

Syntax

gte_lcv2fc()

Explanation

General purpose matrix calculation
 $[lc] * [v2] + [fc]$.

2.9.16. gte_lcirfc

Syntax

gte_lcirfc()

Explanation

General purpose matrix calculation
 $[lc] * [sv] + [fc]$.

2.10. gte_dpcl

Syntax

gte_dpcl()

Explanation

Kernel of DpqColorLight.

2.11. gte_dpccs

Syntax

gte_dpccs()

Explanation

Kernel of DpqColor.

2.12. gte_dpct

Syntax

gte_dpct()

Explanation

Kernel of DpqColor3.

2.13. gte_intpl

Syntax

gte_intpl()

Explanation

Kernel of Intpl.

2.14. gte_sqr12**Syntax**

`gte_sqr12()`

Explanation

Kernel of Square12.

2.15. gte_sqr0**Syntax**

`gte_sqr0()`

Explanation

Kernel of Square0.

2.16. gte_ncs**Syntax**

`gte_ncs()`

Explanation

Kernel of NormalColor.

2.17. gte_nct**Syntax**

`gte_nct()`

Explanation

Kernel of NormalColor3.

2.18. gte_ncds**Syntax**

`gte_ncds()`

Explanation

Kernel of NormalColorDpq.

2.19. gte_ncdt**Syntax**

`gte_ncdt()`

Explanation

Kernel of NormalColorDpq3.

2.20. gte_nccs

Syntax

gte_nccs()

Explanation

Kernel of NormalColorCol.

2.21. gte_ncct

Syntax

gte_ncct()

Explanation

Kernel of NormalColorCol3.

2.22. gte_cdp

Syntax

gte_cdp()

Explanation

Kernel of ColorDpq.

2.23. gte_cc

Syntax

gte_cc()

Explanation

Kernel of ColorCol.

2.24. gte_nclip

Syntax

gte_nclip()

Explanation

Kernel of NormalClip.

2.25. gte_avsz3

Syntax

gte_avsz3()

Explanation

Kernel of AverageZ3.

2.26. gte_avsz4

Syntax

gte_avsz4()

Explanation

Kernel of AverageZ4.

2.27. gte_op12

Syntax

`gte_op12()`

Explanation

Kernel of OuterProduct12.

2.28. gte_op0

Syntax

`gte_op0()`

Explanation

Kernel of OuterProduct0.

2.29. gte_gpf12

Syntax

`gte_gpf12()`

Explanation

First half of LoadAverage12.

2.30. gte_gpf0

Syntax

`gte_gpf0()`

Explanation

First half of LoadAverage0.

2.31. gte_gpl12

Syntax

`gte_gpl12()`

Explanation

Last half of LoadAverage12.

2.32. gte_gpl0

Syntax

`gte_gpl0()`

Explanation

Last half of LoadAverage0.

3. Register Store Functions

3.1. gte_stsxy

Syntax

```
gte_stsxy(sxy)
long *sxy;
```

Explanation

Store screen xy.

3.1.1. gte_stsxy2

Syntax

```
gte_stsxy2(sxy)
long *sxy;
```

Explanation

Store screen xy 2.

3.1.2. gte_stsxy1

Syntax

```
gte_stsxy1(sxy)
long *sxy;
```

Explanation

Store screen xy 1.

3.1.3. gte_stsxy0

Syntax

```
gte_stsxy0(sxy)
long *sxy;
```

Explanation

Store screen xy 0.

3.2. gte_stsxy3

Syntax

```
gte_stsxy3(sxy0,sxy1,sxy2)
long *sxy0, *sxy1, *sxy2;
```

Explanation

Store screen xy 0, 1, 2.

3.2.1. gte_stsxy3_f3

Syntax

```
gte_stsxy3_f3(packet)
u_long *packet
```

Explanation

Store screen xy 0, 1, 2 for POLY_F3.

3.2.2. gte_stsxy3_g3**Syntax**

```
gte_stsxy3_g3(packet)
u_long *packet
```

Explanation

Store screen xy 0, 1, 2 for POLY_G3.

3.2.3. gte_stsxy3_ft3**Syntax**

```
gte_stsxy3_ft3(packet)
u_long *packet
```

Explanation

Store screen xy 0, 1, 2 for POLY_FT3.

3.2.4. gte_stsxy3_gt3**Syntax**

```
gte_stsxy3_gt3(packet)
u_long *packet
```

Explanation

Store screen xy 0, 1, 2 for POLY_GT3.

3.2.5. gte_stsxy3_f4**Syntax**

```
gte_stsxy3_f4(packet)
u_long *packet
```

Explanation

Store screen xy 0, 1, 2 for POLY_F4.

3.2.6. gte_stsxy3_g4**Syntax**

```
gte_stsxy3_g4(packet)
u_long *packet
```

Explanation

Store screen xy 0, 1, 2 for POLY_G4.

3.2.7. gte_stsxy3_ft4**Syntax**

```
gte_stsxy3_ft4(packet)
u_long *packet
```

Explanation

Store screen xy 0, 1, 2 for POLY_FT4.

3.2.8. gte_stsxy3_gt4

Syntax

```
gte_stsxy3_gt4(packet)
u_long *packet
```

Explanation

Store screen xy 0, 1, 2 for POLY_GT4.

3.2.9. gte_stsxy3c

Syntax

```
gte_stsxy3c(sxy)
long sxy[3];
```

Explanation

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

3.2.10. gte_stsxy01

Syntax

```
gte_stsxy01(sxy0,sxy1)
long *sxy0, *sxy1;
```

Explanation

Store screen xy 0, 1.

3.2.11. gte_stsxy01c

Syntax

```
gte_stsxy01c(sxy)
long sxy[2];
```

Explanation

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

3.3. gte_stdp

Syntax

```
gte_stdp(p)
long *p;
```

Explanation

Store depth queuing p.

3.4. gte_stflg

Syntax

```
gte_stflg(flag)
long *flag;
```

Explanation

Store flag.

3.5. gte_stsz**Syntax**

```
gte_stsz(sz)
long *sz;
```

Explanation

Store screen z.

3.6. gte_stsz3**Syntax**

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

Explanation

Store screen z 0, 1, 2.

3.7. gte_stsz4**Syntax**

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

Explanation

Store screen z 0, 1, 2, 3.

3.7.1. gte_stsz4c**Syntax**

```
gte_stsz4c(sz0)
long *sz0;
```

Explanation

Store screen z 0, 1, 2, 3 to continuous addresses.

3.7.2. gte_stsz3c**Syntax**

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

Explanation

Store screen z 0, 1, 2 to continuous addresses.

3.8. gte_stszotz**Syntax**

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

Explanation

Store screen z/4 as OTZ.

3.9. gte_stotz

Syntax

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

Explanation

Store OTZ.

3.10. gte_stopz

Syntax

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

Explanation

Store outer product.

3.11. gte_stlvl

Syntax

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

Explanation

Store VECTOR from 16 bit universal register.

3.12. gte_stlvnl

Syntax

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

Explanation

Store VECTOR from 32 bit universal register.

3.12.1. gte_stlvnl0

Syntax

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

Explanation

Store 1st component from 32 bit universal register.

3.12.2. gte_stlvnl1

Syntax

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

Explanation

Store 2nd component from 32 bit universal register.

3.12.3. gte_stlwnl2**Syntax**

```
gte_stlwnl2(x)
long *x;
```

Explanation

Store 3rd component from 32 bit universal register.

3.13. gte_stsv**Syntax**

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

Explanation

Store SVECTOR from 16 bit universal register.

3.14. gte_stclmv**Syntax**

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

Explanation

Store MATRIX column from 16 bit universal register.

3.15. gte_stbv**Syntax**

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

Explanation

Store byte vector from LS 8 bits of 16 bit universal register.

3.16. gte_stcv**Syntax**

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

Explanation

Store CVECTOR from LS 8 bits of 16 bit universal register.

3.17. gte_strgb**Syntax**

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

Explanation

Store CVECTOR from color register.

3.18. gte_strgb3

Syntax

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

Explanation

Store CVECTOR 0, 1, 2 from color fifo.

3.18.1. gte_strgb3_g3

Syntax

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

Explanation

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

3.18.2. gte_strgb3_gt3

Syntax

```
gte_strgb3_gt3(packet)
u_long *packet
```

Explanation

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

3.18.3. gte_strgb3_g4

Syntax

```
gte_strgb3_g4(packet)
u_long *packet
```

Explanation

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

3.18.4. gte_strgb3_gt4

Syntax

```
gte_strgb3_gt4(packet)
u_long *packet
```

Explanation

Store CVECTOR 0, 1, 2 from color fifo to POLY_GT4 packet.

3.19. gte_ReadGeomOffset

Syntax

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

Explanation

Store GTE-offset.

3.20. gte_ReadGeomScreen**Syntax**

```
gte_ReadGeomScreen(h)
long *h;
```

Explanation

Store distance from viewpoint to screen.

3.21. gte_ReadRotMatrix**Syntax**

```
gte_ReadRotMatrix(m)
MATRIX *m;
```

Explanation

Store Rotation Matrix.

3.21.1. gte_sttr**Syntax**

```
gte_sttr(v)
VECTOR *v;
```

Explanation

Store Transfer Vector.

3.22. gte_ReadLightMatrix**Syntax**

```
gte_ReadLightMatrix(m)
MATRIX *m;
```

Explanation

Store Light Matrix.

3.23. gte_ReadColorMatrix**Syntax**

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

Explanation

Store Color Matrix.

3.24. gte_stlzc**Syntax**

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

Explanation

Store LZC.

3.25. gte_stfc

Syntax

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

Explanation

Store far color.

4. Register Move Functions

4.1. gte_mvlvtr

Syntax

```
gte_mvlvtr()
```

Explanation

Move 32 bit universal vector to Transfer Vector.

5. Miscellaneous

5.1. gte_nop

Syntax

```
gte_nop()
```

Explanation

No operation.

5.2. gte_subdvl

Syntax

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

Explanation

$v3 = v1 - v2$.

5.3. gte_subdvd

Syntax

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

Explanation

$v3 = v1 - v2$.

5.4. gte_adddvl

Syntax

```
gte_adddvl(v1,v2,v3)
DVECTOR *v1, *v2
VECTOR *v3
```

Explanation

$v3 = v1 + v2$.

5.5. gte_adddvd

Syntax

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

Explanation

$v3 = v1 + v2$.

5.6. gte_FlipRotMatrixX

Syntax

```
gte_FlipRotMatrixX()
```

Explanation

Flip X-row of Rotation Matrix.
 $(R11, R12, R13) \rightarrow (-R11, -R12, -R13)$

5.6.1. gte_FlipTRX

Syntax

```
gte_FlipTRX()
```

Explanation

Flip X of transfer vector.
 $TRX \rightarrow -TRX$

Chapter 3:

GTE Inline Macros

1. Simple Functions

1.1. gte_RotTransPers

Syntax

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

Explanation

*r5 is the return value of RotTransPers().

1.2. gte_RotTransPers3

Syntax

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

Explanation

*r9 is the return value of RotTransPers3().

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.

1.11. gte_Square0

Syntax

```
gte_Square0(r1,r2)
```

Explanation

No return value.

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 the return value of NormalClip().

1.21. gte_AverageZ3**Syntax**

```
gte_AverageZ3(r1,r2,r3,r4)
```

Explanation

*r4 is the return value of AverageZ3().

1.22. gte_AverageZ4**Syntax**

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

Explanation

*r5 is the return value of AverageZ4().

1.23. gte_OuterProduct12**Syntax**

```
gte_OuterProduct12(r1,r2,r3)
```

Explanation

Warning! Use of this function will destroy the Rotation Matrix in GTE.
(Note that the original function, OuterProduct12, will not.)

1.24. gte_OuterProduct0**Syntax**

```
gte_OuterProduct0(r1,r2,r3)
```

Explanation

Warning! Use of this function will destroy the Rotation Matrix in GTE.
(Note that the original function, OuterProduct0, will not.)

1.25. gte_Lzc**Syntax**

```
gte_Lzc(r1,r2)
```

Explanation

*r2 is the return value of Lzc().

2. Combined Functions

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

2.1. gte_RotAverage3**Syntax**

```
gte_RotAverage3(r1,r2,r3,r4,r5,r6,r7,r8,r9)
```

Explanation

*r9 is the return value of RotAverage3().

2.2. gte_RotNclip3**Syntax**

```
gte_RotNclip3(r1,r2,r3,r4,r5,r6,r7,r8,r9,r10)
```

Explanation

*r10 is the return value of RotNclip3().

2.3. gte_RotAverageNclip3**Syntax**

```
gte_RotAverageNclip3(r1,r2,r3,r4,r5,r6,r7,r8,r9,r10)
```

Explanation

*r10 is the return value of RotAverageNclip3().

2.4. gte_RotColorDpq**Syntax**

```
gte_RotColorDpq(r1,r2,r3,r4,r5,r6,r7)
```

Explanation

*r7 is the return value of RotColorDpq().

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 the return value of RotColorDpq3().

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 the return value of RotAverageNclipColorDpq3().

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 the return value of RotAverageNclipColorCol3().

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

Warning! Use of this function will destroy the Rotation Matrix in GTE.
(Note that the original function, MulMatrix0, will also destroy the Rotation Matrix.)

3.2. gte_ApplyMatrix

Syntax

```
gte_ApplyMatrix(r1,r2,r3)
```

Explanation

Warning! Use of this function will destroy the Rotation Matrix in GTE.

(Note that the original function, ApplyMatrix will also destroy the Rotation Matrix.)

3.3. gte_CompMatrix**Syntax**

`gte_CompMatrix(r1,r2,r3)`

Explanation

Warning! Use of this function will destroy the Rotation Matrix in GTE.

(Note that the original function, CompMatrix will also destroy the Rotation Matrix.)

Warning! Use of this function will destroy the Transfer Vector in GTE.

(Note that the original function, CompMatrix, will not.)

3.4. gte_ApplyRotMatrix**Syntax**

`gte_ApplyRotMatrix(r1,r2)`

Explanation

Chapter 4:

GTE Programming Guide

Format

GTE is a vector/matrix processor implemented as "coprocessor 2" under the MIPS architecture specification. The data format it handles consists of fixed decimal (fractional) real numbers.

Registers

Coprocessors in the MIPS architecture can have two sets of registers: "data registers" and "control registers". GTE has 32 data registers and 32 control registers. These are all 32-bit registers, and access by the CPU is performed in register units. However, some registers are divided into multiple 8- or 16-bit fields. Only GTE registers are referenced or changed when GTE is performing calculations.

For a detailed description of the various registers, refer to the "GTE Register Specification".

Register Access Instructions

Data is transferred between GTE registers and CPU registers, or GTE registers and main memory (including the scratchpad) by executing the CPU instructions listed below.

CPU Instruction	Source	Destination
lwc2	Memory scratchpad	GTE data register
swc2	GTE data register	Memory scratchpad
mtc2	CPU general-purpose register	GTE data register
mfc2	GTE data register	CPU general-purpose register
ctc2	CPU general-purpose register	GTE control register
cfc2	GTE control register	CPU general-purpose register

The instructions mtc2, mfc2, ctc2, and cfc2 transfer data between registers. However, as is the case for on-cache memory accesses, delayed loads should be used. In examples such as those listed below, nop should be inserted into the delay slot.

Example 1:

```
cfc2    v0,C2_FLAG
and     v0,v0,v1      # No good
```

Example 2:

```
cfc2    v0,C2_FLAG
nop
and     v0,v0,v1      # delay slot
```

Register Names

The macro definition of GTE register names can be found in the include header file "gtereg.h", which is part of the PlayStation library. The names of the macros are formed by adding the prefix "C2_" to the register names used in the "GTE Register Specification".

Register number	Data register	Control register
0	C2_VXY0	C2_R11R12
1	C2_VZ0	C2_R13R21
2	C2_VXY1	C2_R22R23
3	C2_VZ1	C2_R31R32
4	C2_VXY2	C2_R33
5	C2_VZ2	C2_TRX
6	C2_RGB	C2_TRY
7	C2_OTZ	C2_TRZ
8	C2_IR0	C2_L11L12
9	C2_IR1	C2_L13L21
10	C2_IR2	C2_L22L23
11	C2_IR3	C2_L31L32
12	C2_SXY0	C2_L33
13	C2_SXY1	C2_RBK
14	C2_SXY2	C2_GBK
15	C2_SXYP	C2_BBK
16	C2_SZ0	C2_LR1LR2
17	C2_SZ1	C2_LR3LG1
18	C2_SZ2	C2_LG2LG3
19	C2_SZ3	C2_LB1LB2
20	C2_RGB0	C2_LB3
21	C2_RGB1	C2_RFC
22	C2_RGB2	C2_GFC
23	Undefined	C2_BFC
24	C2_MAC0	C2_OFX
25	C2_MAC1	C2_OFY
26	C2_MAC2	C2_H
27	C2_MAC3	C2_DQA
28	C2_IRGB	C2_DQB
29	C2_ORGB	C2_ZSF3
30	C2_LZCS	C2_ZSF4
31	C2_LZCR	C2_FLAG

Commands

GTE can perform an entire series of calculations essential for graphics programming (such as coordinate transformation, perspective transformation, and light source calculation) by executing a single command. Also, general-purpose matrix and vector calculations (such as matrix calculation, outer product, and interpolation) are available as commands. In all of the above cases, the calculation speed is several times faster than if the calculations in question were performed by the CPU.

Refer to the "GTE Command Reference" for a detailed description of the available GTE commands. Also, GTE commands are macro defined in the file "inline.h", which is included with DMPSX.

Delay Slots

CPU instructions that execute coprocessor 2 commands (referred to as "cop2"), require two delay slots for preceding GTE-related instructions.

Example 3:


```
mtc2   v0,C2_VXY0
nop                    # delay slot
nop                    # delay slot
RTPS
```

Command Execution Cycles

The various GTE commands require the number of cycles listed below to complete. After a coprocessor 2 execute instruction is issued, if the prescribed number of cycles is not left open, and either a GTE register read instruction (swc2, mfc2, cfc2) or another coprocessor 2 command execute instruction is issued, the CPU will stall until the initial coprocessor 2 instruction has completed execution.


Example 4:

```
RTPS
## interlock
cfc2   v0,C2_FLAG
```



Example 5:

```
RTPS
add    v1,v2,v3
sub    v1,v2,v3
## interlock
cfc2   v0,C2_FLAG
```



Coding Limitations

- [1] A GTE instruction (cop2) must not be executed in an event handler or callback function.
- [2] A GTE instruction (cop2) must not be inserted into the delay slot following a jump or branch instruction.
- [3] A GTE register access instruction (lwc2, swc2, mtc2, mfc2, ctc2, cfc2) must not be inserted into the delay slot following a jump or branch command.
- [4] A GTE register load instruction (lwc2, mtc2, ctc2) must not be used between a GTE instruction (cop2) and a GTE register save instruction (swc2, mfc2, cfc2) or between a GTE instruction (cop2) and another GTE instruction (cop2).

Example 6:

```
/* cop2-load-save (NG) */
RTPS                                /* cop2 */
                                   /* cpu instructions */
mtc2  v0,C2_VXY0                   /* NG !!!!! */
                                   /* cpu instructions */
cfc2   v0,C2_FLAG                   /* save instruction */
```

Example 7:

```
/* cop2-load-cop2 (NG) */
RTPT                                /* cop2 */
                                   /* cpu instructions */
mtc2  v0,C2_VXY0                   /* NG !!!!! */
                                   /* cpu instructions */
NCLIP                                /* cop2 */
```

[5] If a GTE register to which data is to be loaded is not being referenced or overwritten by a GTE command that is currently executing, it is possible to execute a command which transfers data to the GTE register without worrying about the GTE command (cop2).

Example 8:

```
/* cop2-load-save (OK) */
RTPS                                /* cop2 */
                                   /* cpu instructions */
mtc2  v0,C2_VXY1                   /* OK !! */
                                   /* cpu instructions */
cfc2   v0,C2_FLAG                   /* save instruction */
```

Example 9:

```
/* cop2-load-cop2 (OK) */
RTPT                                /* cop2 */
                                   /* cpu instructions */
mtc2  v0,C2_RGB                    /* OK !! */
                                   /* cpu instructions */
NCLIP                                /* cop2 */
```

Recommended Development Style

When coding in assembler, programs that violate some of the above rules may appear to run properly at first glance. However, such violations tend to become evident as bugs that are extremely difficult to track down, such as incorrect operation during an interrupt. For this reason, programmers are advised to avoid coding directly in assembler as much as possible.

To prevent bad code from being generated, the following development sequence should be used.

C (libgte) -> C (DMPSX) -> Assembler (DMPSX)

Chapter 5:

GTE Register Specification

Control Register Group

Register number	Name	Access	Content
0	R11R12	R/W	Rotation matrix
1	R13R21	R/W	Rotation matrix
2	R22R23	R/W	Rotation matrix
3	R31R32	R/W	Rotation matrix
4	R33	R/W	Rotation matrix
5	TRX	R/W	Translation vector (X)
6	TRY	R/W	Translation vector (Y)
7	TRZ	R/W	Translation vector (Z)
8	L11L12	R/W	Light source direction vector X 3
9	L13L21	R/W	Light source direction vector X 3
10	L22L23	R/W	Light source direction vector X 3
11	L31L32	R/W	Light source direction vector X 3
12	L33	R/W	Light source direction vector X 3
13	RBK	R/W	Peripheral color (background color) (R)
14	GBK	R/W	Peripheral color (background color) (G)
15	BBK	R/W	Peripheral color (background color) (B)
16	LR1LR2	R/W	Light source color X 3
17	LR3LG1	R/W	Light source color X 3
18	LG2LG3	R/W	Light source color X 3
19	LB1LB2	R/W	Light source color X 3
20	LB3	R/W	Light source color X 3
21	RFC	R/W	Far color (R)
22	GFC	R/W	Far color (G)
23	BFC	R/W	Far color (B)
24	OFX	R/W	Screen offset (X)
25	OFY	R/W	Screen offset (Y)
26	H	R/W	Screen position
27	DQA	R/W	Depth parameter A (coefficient)
28	DQB	R/W	Depth parameter B (offset)
29	ZSF3	R/W	Z-averaging scale factor
30	ZSF4	R/W	Z-averaging scale factor
31	FLAG	R	Flag

Data Register Group

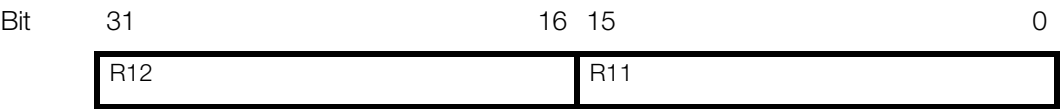
Register number	Name	Access	Content
0	VXY0	R/W	Vector #0 (X/Y)
1	VZ0	R/W	Vector #0 (Z)
2	VXY1	R/W	Vector #1 (X/Y)
3	VZ1	R/W	Vector #1 (Z)
4	VXY2	R/W	Vector #2 (X/Y)
5	VZ2	R/W	Vector #2 (Z)
6	RGB	R/W	Color data + GTE instruction
7	OTZ	R	Z-component average value
8	IR0	R/W	Intermediate value #0
9	IR1	R/W	Intermediate value #1
10	IR2	R/W	Intermediate value #2
11	IR3	R/W	Intermediate value #3
12	SXY0	R/W	Calculation result record (XY)
13	SXY1	R/W	Calculation result record (XY)
14	SXY2	R/W	Calculation result record (XY)
15	 SXYP	W	Calculation result setting register
16	SZ0	R/W	Calculation result record (Z)
17	SZ1	R/W	Calculation result record (Z)
18	SZ2	R/W	Calculation result record (Z)
19	SZ3	R/W	Calculation result record (Z)
20	RGB0	R/W	Calculation result record (color data)
21	RGB1	R/W	Calculation result record (color data)
22	RGB2	R/W	Calculation result record (color data)
23	RES1	n/a	Reserved by system (access prohibited)
24	MAC0	R	Sum of products #0
25	MAC1	R/W	Sum of products #1
26	MAC2	R/W	Sum of products #2
27	MAC3	R/W	Sum of products #3
28	IRGB	W	Color data input register
29	ORGB	R	Color data output register
30	LZCS	W	Leading zero/one count source data
31	LZCR	R	Leading zero/one count processing result

Register number: Control #0

Register name: R11R12

Access: R/W

Bit pattern:



Fields:

R11 (1.3.12) Element (1,1) of rotation matrix

R12 (1.3.12) Element (1,2) of rotation matrix

Matrix expression:

Matrix X =
$$\begin{bmatrix} (1,1), (1,2), (1,3) \\ (2,1), (2,2), (2,3) \\ (3,1), (3,2), (3,3) \end{bmatrix}$$

Register number: Control #1

Register name: R21R13

Access: R/W

Bit pattern:

Bit	31	16	15	0
-----	----	----	----	---



Fields:

R13 (1.3.12) Element (1,3) of rotation matrix

R21	(1.3.12)	Element (2,1) of rotation matrix
-----	----------	----------------------------------

Matrix expression:

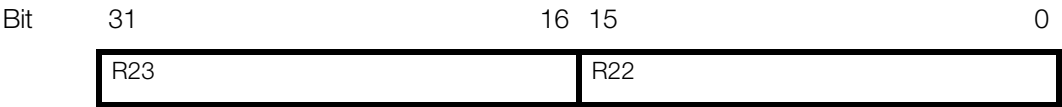
$$\text{Matrix X} = \begin{bmatrix} (1,1), (1,2), (1,3) \\ (2,1), (2,2), (2,3) \\ (3,1), (3,2), (3,3) \end{bmatrix}$$

Register number: Control #2

Register name: R23R22

Access: R/W

Bit pattern:



Fields:

- | | | |
|-----|----------|----------------------------------|
| R22 | (1.3.12) | Element (2,2) of rotation matrix |
| R23 | (1.3.12) | Element (2,3) of rotation matrix |

Matrix expression:

Matrix X =
$$\begin{bmatrix} (1,1), (1,2), (1,3) \\ (2,1), (2,2), (2,3) \\ (3,1), (3,2), (3,3) \end{bmatrix}$$

Register number: Control #3

Register name: R32R31

Access: R/W

Bit pattern:

Bit 31 16 15 0



Fields:

R31	(1.3.12)	Element (3,1) of rotation matrix
-----	----------	----------------------------------

R32	(1.3.12)	Element (3,2) of rotation matrix
-----	----------	----------------------------------

Matrix expression:

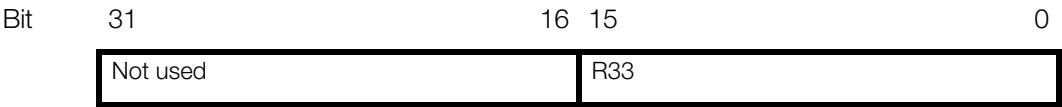
$$\text{Matrix X} = \begin{bmatrix} (1,1), (1,2), (1,3) \\ (2,1), (2,2), (2,3) \\ (3,1), (3,2), (3,3) \end{bmatrix}$$

Register number: Control #4

Register name: R33

Access: R/W

Bit pattern:



Fields:

R33 (1.3.12) Element (3,3) of rotation matrix

Matrix expression:

Matrix X =
$$\begin{bmatrix} (1,1), (1,2), (1,3) \\ (2,1), (2,2), (2,3) \\ (3,1), (3,2), (3,3) \end{bmatrix}$$

Register number: Control #5

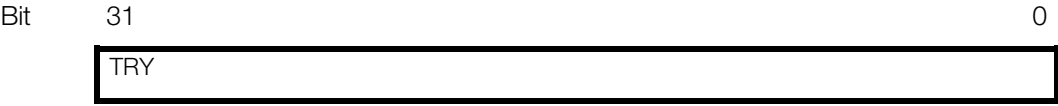
Register name: TRX
Access: R/W
Bit pattern:



Fields:
TRX (1.31.0) Translation vector X-component

Register number: Control #6

Register name: TRY
Access: R/W
Bit pattern:



Fields:
TRY (1.31.0) Translation vector Y-component

Register number: Control #7

Register name: TRZ

Access: R/W

Bit pattern:



Fields:

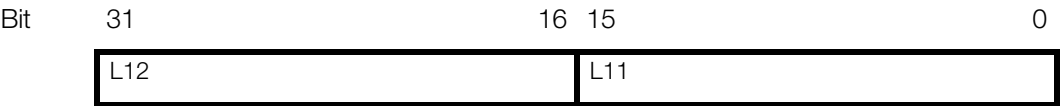
TRZ (1.31.0) Translation vector Z-component

Register number: Control #8

Register name: L11L12

Access: R/W

Bit pattern:



Fields:

L11 (1.3.12) Light source direction vector #1 X-component

L12 (1.3.12) Light source direction vector #1 Y-component

Matrix expression:

"Light source direction vector X 3" is a matrix combining three light source direction vectors. The allocation of the elements is as follows.

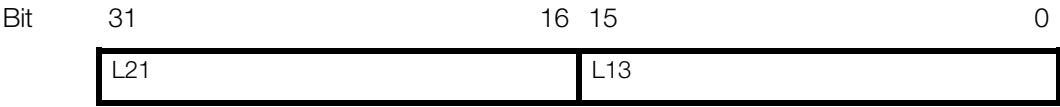
$$\text{Matrix} = \begin{bmatrix} (1, X), (1, Y), (1, Z) \\ (2, X), (2, Y), (2, Z) \\ (3, X), (3, Y), (3, Z) \end{bmatrix}$$

Register number: Control #9

Register name: L21L13

Access: R/W

Bit pattern:



Fields:

- L13

(1.3.12)

Light source direction vector #1 Z-component
- L21

(1.3.12)

Light source direction vector #2 X-component

Matrix expression:

"Light source direction vector X 3" is a matrix combining three light source direction vectors. The allocation of the elements is as follows.

Matrix=

⌈

(1,X),(1,Y),(1,Z)

⌋

⌈

(2,X),(2,Y),(2,Z)

⌋

⌈

(3,X),(3,Y),(3,Z)

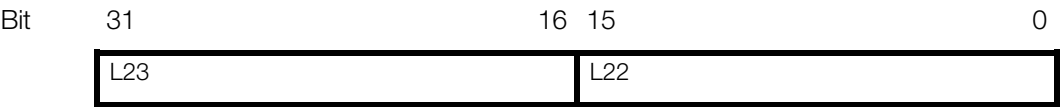
⌋

Register number: Control #10

Register name: L23L22

Access: R/W

Bit pattern:



Fields:

- | | | |
|-----|----------|--|
| L22 | (1.3.12) | Light source direction vector #2 Y-component |
| L23 | (1.3.12) | Light source direction vector #2 Z-component |

Matrix expression:

"Light source direction vector X 3" is a matrix combining three light source direction vectors. The allocation of the elements is as follows.

Matrix=

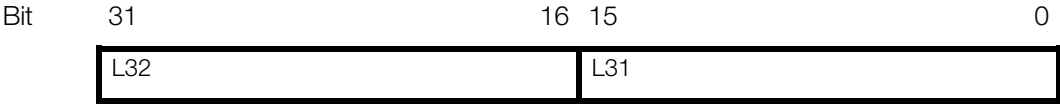
$$\begin{bmatrix} (1,X), (1,Y), (1,Z) \\ (2,X), (2,Y), (2,Z) \\ (3,X), (3,Y), (3,Z) \end{bmatrix}$$

Register number: Control #11

Register name: L32L31

Access: R/W

Bit pattern:



Fields:

L31	(1.3.12)	Light source direction vector #3 X-component
L32	(1.3.12)	Light source direction vector #3 Y-component

Matrix expression:

"Light source direction vector X 3" is a matrix combining three light source direction vectors. The allocation of the elements is as follows.

Matrix=

(1,X)

(1,Y)

(1,Z)

(2,X)

(2,Y)

(2,Z)

(3,X)

(3,Y)

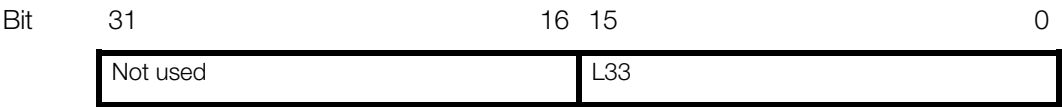
(3,Z)

Register number: Control #12

Register name: L33

Access: R/W

Bit pattern:



Fields:

L33 (1.3.12) Light source direction vector #3 Z-component

Matrix expression:

"Light source direction vector X 3" is a matrix combining three light source direction vectors. The allocation of the elements is as follows.

Matrix=
$$\begin{bmatrix} (1,X), (1,Y), (1,Z) \\ (2,X), (2,Y), (2,Z) \\ (3,X), (3,Y), (3,Z) \end{bmatrix}$$

Register number: Control #13

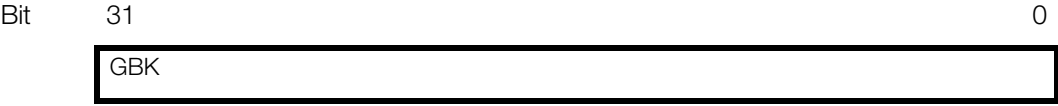
Register name: RBK
 Access: R/W
 Bit pattern:



Fields:
 RBK (1.19.12) Background color R-component

Register number: Control #14

Register name: GBK
 Access: R/W
 Bit pattern:



Fields:
 GBK (1.19.12) Background color G-component

Register number: Control #15

Register name: BBK

Access: R/W

Bit pattern:

Bit	31	0
-----	----	---



Fields:

BBK	(1.19.12)	Background color B-component
-----	-----------	------------------------------

Register number: Control #16

Register name: LR1LR2

Access: R/W

Bit pattern:

Bit	31	16	15	0
-----	----	----	----	---



Fields:

LR1	(1.3.12)	Light source color #1 R-component
-----	----------	-----------------------------------

LR2	(1.3.12)	Light source color #2 R-component
-----	----------	-----------------------------------

Matrix expression:

"Light source color X 3" is a matrix combining three light source RGB expression color data values. The allocation of the elements is as follows.

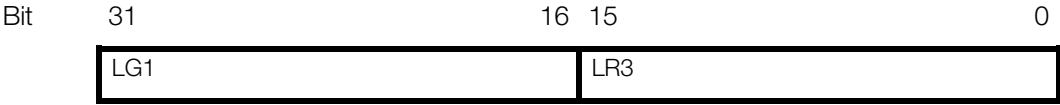
$$\text{Matrix} = \begin{bmatrix} (R,1), (R,2), (R,3) \\ (G,1), (G,2), (G,3) \\ (B,1), (B,2), (B,3) \end{bmatrix}$$

Register number: Control #17

Register name: LR3LG1

Access: R/W

Bit pattern:



Fields:

LR3 (1.3.12) Light source color #3 R-component

LG1 (1.3.12) Light source color #1 G-component

Matrix expression:

"Light source color X 3" is a matrix combining three light source RGB expression color data values. The allocation of the elements is as follows.

Matrix=

⌈

(R,1),(R,2),(R,3)

⌋

(G,1),(G,2),(G,3)

⌋

⌈

(B,1),(B,2),(B,3)

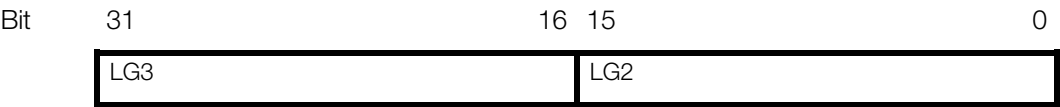
⌋

Register number: Control #18

Register name: LG2LG3

Access: R/W

Bit pattern:



Fields:

- | | | |
|-----|----------|-----------------------------------|
| LG2 | (1.3.12) | Light source color #2 G-component |
| LG3 | (1.3.12) | Light source color #3 G-component |

Matrix expression:

"Light source color X 3" is a matrix combining three light source RGB expression color data values. The allocation of the elements is as follows.

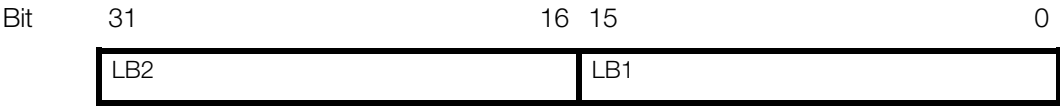
Matrix=
$$\begin{bmatrix} (R,1), (R,2), (R,3) \\ (G,1), (G,2), (G,3) \\ (B,1), (B,2), (B,3) \end{bmatrix}$$

Register number: Control #19

Register name: LB1LB2

Access: R/W

Bit pattern:



Fields:

- LB1

(1.3.12)

Light source color #1 B-component
- LB2

(1.3.12)

Light source color #2 B-component

Matrix expression:

"Light source color X 3" is a matrix combining three light source RGB expression color data values. The allocation of the elements is as follows.

Matrix=

⌈

(R,1),(R,2),(R,3)

⌋

(G,1),(G,2),(G,3)

⌋

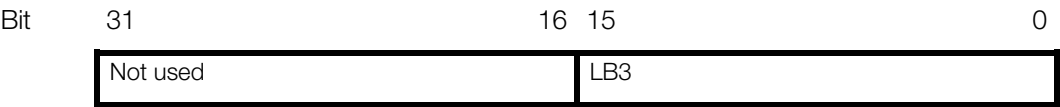
⌈

(B,1),(B,2),(B,3)

⌋

Register number: Control #20

Register name: LB3
Access: R/W
Bit pattern:



Fields:
LB3 (1.3.12) Light source color #3 B-component

Matrix expression:
"Light source color X 3" is a matrix combining three light source RGB expression color data values. The allocation of the elements is as follows.

Matrix=
$$\begin{bmatrix} (R,1), (R,2), (R,3) \\ (G,1), (G,2), (G,3) \\ (B,1), (B,2), (B,3) \end{bmatrix}$$

Register number: Control #21

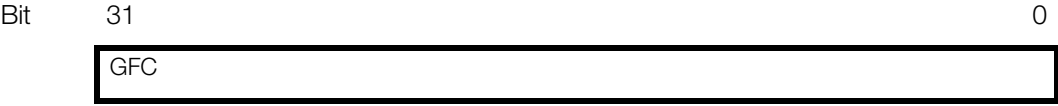
Register name: RFC
 Access: R/W
 Bit pattern:



Fields:
 RFC (1.27.4) Far color R-component

Register number: Control #22

Register name: GFC
 Access: R/W
 Bit pattern:



Fields:
 GFC (1.27.4) Far color G-component

Register number: Control #23

Register name: BFC

Access: R/W

Bit pattern:

Bit 31 0



Fields:

BFC	(1.27.4)	Far color B-component
-----	----------	-----------------------

Register number: Control #24

Register name: OFX

Access: R/W

Bit pattern:

Bit	31	0
-----	----	---



Fields:

OFX	(1.15.16)	Screen offset X-component
-----	-----------	---------------------------

Register number: Control #25

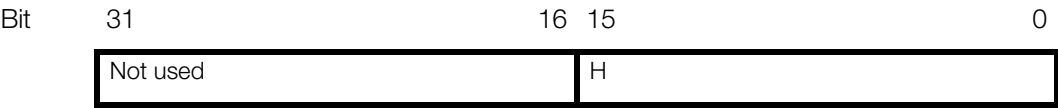
Register name: OFY
 Access: R/W
 Bit pattern:



Fields:
 OFY (1.15.16) Screen offset Y-component

Register number: Control #26

Register name: H
 Access: R/W
 Bit pattern:



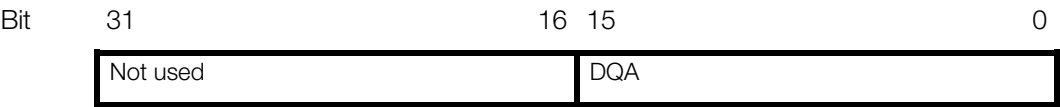
Fields:
 H (0.16.0) Screen position

Register number: Control #27

Register name: DQA

Access: R/W

Bit pattern:



Fields:

DQA (1.7.8) Depth parameter A (coefficient)

Register number: Control #28

Register name: DQB

Access: R/W

Bit pattern:

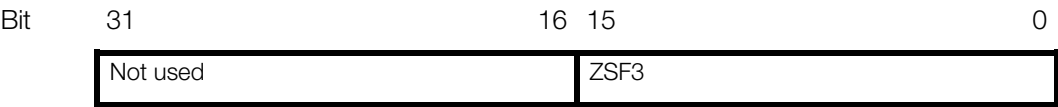


Fields:

DQB (1.7.24) Depth parameter B (offset)

Register number: Control #29

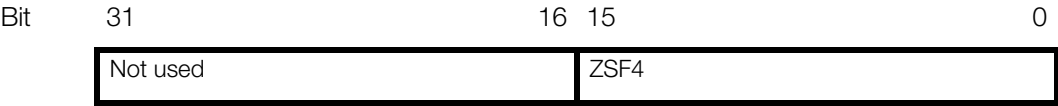
Register name: ZSF3
 Access: R/W
 Bit pattern:



Fields:
 ZSF3 (1.3.12) Z-averaging scale factor (normally set to 1/3)

Register number: Control #30

Register name: ZSF4
 Access: R/W
 Bit pattern:



Fields:
 ZSF4 (1.3.12) Z-averaging scale factor (normally set to 1/4)

Register number: Control #31

Register name: FLAG

Access: R/W

Bit pattern:

Bit	31	12	11	0
	FLAG			Not used

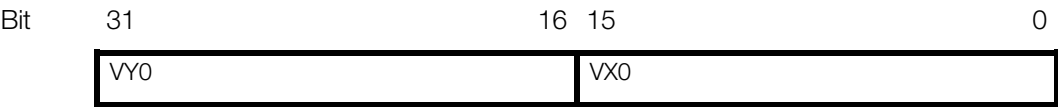
Fields:

FLAG As indicated in table below

Bit number	Content
31	Logical sum of bits 30 - 23 and bits 18 - 13
30	1: Calculation test result #1 overflow generated (2^{43} or more)
29	1: Calculation test result #2 overflow generated (2^{43} or more)
28	1: Calculation test result #3 overflow generated (2^{43} or more)
27	1: Calculation test result #1 underflow generated (less than -2^{43})
26	1: Calculation test result #2 underflow generated (less than -2^{43})
25	1: Calculation test result #3 underflow generated (less than -2^{43})
24	1: Limiter A1 out of range detected (less than 0 or less than -2^{15} , or 2^{15} or more)
23	1: Limiter A2 out of range detected (less than 0 or less than -2^{15} , or 2^{15} or more)
22	1: Limiter A3 out of range detected (less than 0 or less than -2^{15} , or 2^{15} or more)
21	1: Limiter B1 out of range detected (less than 0, or 2^8 or more)
20	1: Limiter B2 out of range detected (less than 0, or 2^8 or more)
19	1: Limiter B3 out of range detected (less than 0, or 2^8 or more)
18	1: Limiter C out of range detected (less than 0, or 2^{16} or more)
17	1: Divide overflow generated (quotient of 2.0 or more)
16	1: Calculation test result #4 overflow generated (2^{31} or more)
15	1: Calculation test result #4 underflow generated (less than -2^{31})
14	1: Limiter D1 out of range detected (less than -2^{10} , or 2^{10} or more)
13	1: Limiter D2 out of range detected (less than -2^{10} , or 2^{10} or more)
12	1: Limiter E out of range detected (less than 0, or 2^{12} or more)

Register number: Data #0

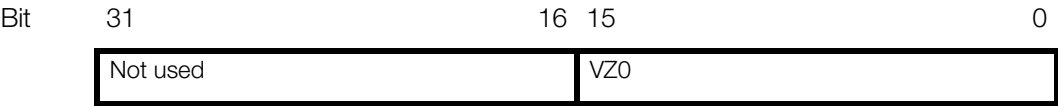
Register name: VXY0
 Access: R/W
 Bit pattern:



Fields:
 VX0 (1.15.0) or (1.3.12) Vector #0 X-element
 VY0 (1.15.0) or (1.3.12) Vector #0 Y-element

Register number: Data #1

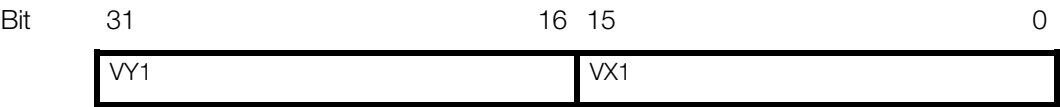
Register name: VZ0
 Access: R/W
 Bit pattern:



Fields:
 VZ0 (1.15.0) or (1.3.12) Vector #0 Z-element

Register number: Data #2

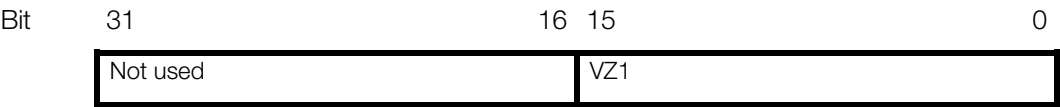
Register name: VXY1
Access: R/W
Bit pattern:



Fields:
VX1 (1.15.0) or (1.3.12) Vector #1 X-element
VY1 (1.15.0) or (1.3.12) Vector #1 Y-element

Register number: Data #3

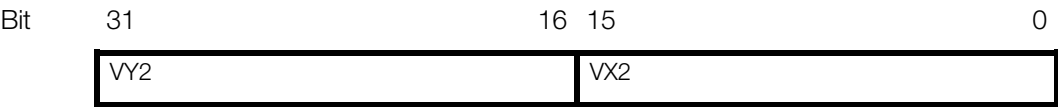
Register name: VZ1
Access: R/W
Bit pattern:



Fields:
VZ1 (1.15.0) or (1.3.12) Vector #1 Z-element

Register number: Data #4

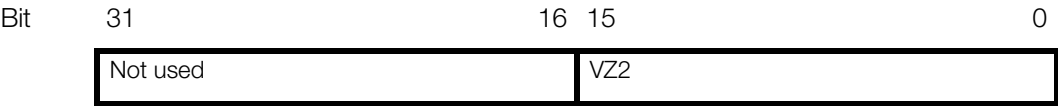
Register name: VXY2
 Access: R/W
 Bit pattern:



Fields:
 VX2 (1.15.0) or (1.3.12) Vector #2 X-element
 VY2 (1.15.0) or (1.3.12) Vector #2 Y-element

Register number: Data #5

Register name: VZ2
 Access: R/W
 Bit pattern:



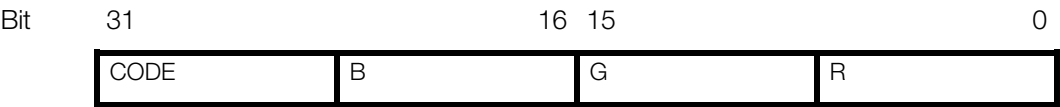
Fields:
 VZ2 (1.15.0) or (1.3.12) Vector #2 Z-element

Register number: Data #6

Register name: RGB

Access: R/W

Bit pattern:



Fields:

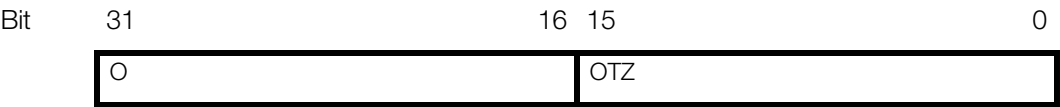
R	(0.8.0)	Characteristic color R-element
G	(0.8.0)	Characteristic color G-element
B	(0.8.0)	Characteristic color B-element
CODE	(-.8.-)	Arbitrary 8-bit data (normally specified by GPU draw command)

Register number: Data #7

Register name: OTZ

Access: R

Bit pattern:

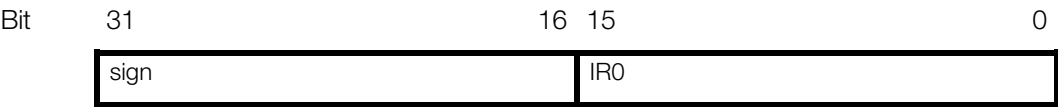


Fields:

OTZ	(0.15.0)	Z-element average value
-----	----------	-------------------------

Register number: Data #8

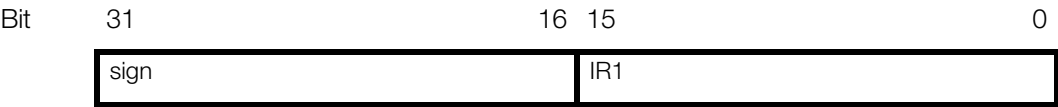
Register name: IR0
 Access: R/W
 Bit pattern:



Fields:
 IR0 (1.3.12) or the like Intermediate value #0
 sign All bits 0 or 1

Register number: Data #9

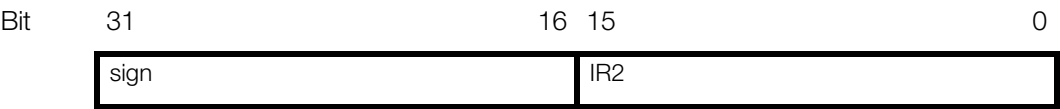
Register name: IR1
 Access: R/W
 Bit pattern:



Fields:
 IR1 (1.3.12) or the like Intermediate value #1
 sign All bits 0 or 1

Register number: Data #10

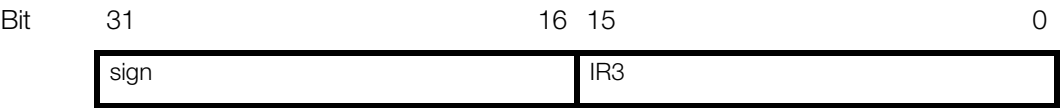
Register name: IR2
Access: R/W
Bit pattern:



Fields:
IR2 (1.3.12) or the like Intermediate value #2
sign All bits 0 or 1

Register number: Data #11

Register name: IR3
Access: R/W
Bit pattern:



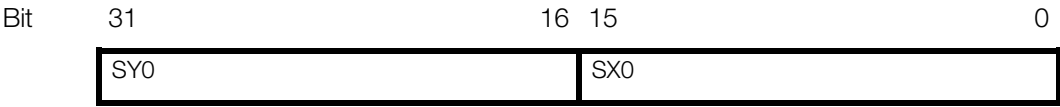
Fields:
IR3 (1.3.12) or the like Intermediate value #3
sign All bits 0 or 1

Register number: Data #12

Register name: SXY0

Access: R/W

Bit pattern:



Fields:		
SX0	(1.15.0)	X-element of 2-dimensional screen coordinates or 2-dimensional coordinates following perspective transformation. Note that this value was obtained in the calculation three times previous.
SY0	(1.15.0)	Y-element of 2-dimensional screen coordinates or 2-dimensional coordinates following perspective transformation. Note that this value was obtained in the calculation three times previous.

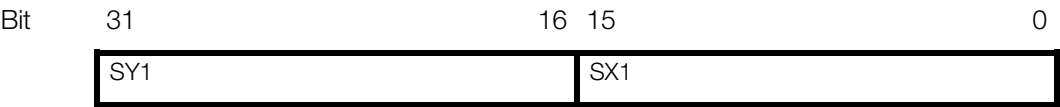
Internal operations:
See Data #14: SXY2 and Data #15: SXYP.

Register number: Data #13

Register name: SXY1

Access: R/W

Bit pattern:



Fields:

SX1	(1.15.0)	X-element of 2-dimensional screen coordinates or 2-dimensional coordinates following perspective transformation. Note that this value was obtained in the calculation two times previous.
SY1	(1.15.0)	Y-element of 2-dimensional screen coordinates or 2-dimensional coordinates following perspective transformation. Note that this value was obtained in the calculation two times previous.

Internal operations:

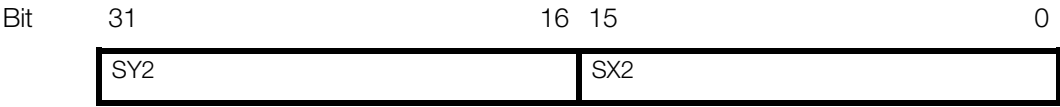
See Data #14: SXY2 and Data #15: SXYP.

Register number: Data #14

Register name: SXY2

Access: R/W

Bit pattern:



Fields:		
SX2	(1.15.0)	X-element of 2-dimensional screen coordinates or 2-dimensional coordinates following perspective transformation. Note that this value was obtained in the calculation one time previous.
SY2	(1.15.0)	Y-element of 2-dimensional screen coordinates or 2-dimensional coordinates following perspective transformation. Note that this value was obtained in the calculation one time previous.

Internal operations:

In several GTE instructions, substitutions are made in the following sequence.

SXY0 = SXY1;

SXY1 = SXY2;

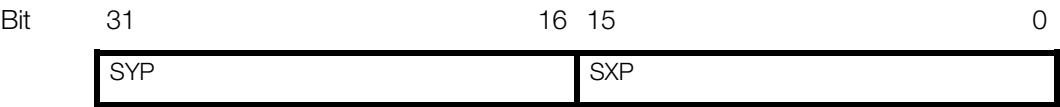
SXY2 = Coordinate XY-elements obtained through calculation.

Register number: Data #15

Register name: SXYP

Access: W

Bit pattern:



Fields:

SXP	(1.15.0)	X-element of coordinates transferred to SXY2
SYP	(1.15.0)	Y-element of coordinates transferred to SXY2

Internal operations:

The following operations are generated at the same time as the write.

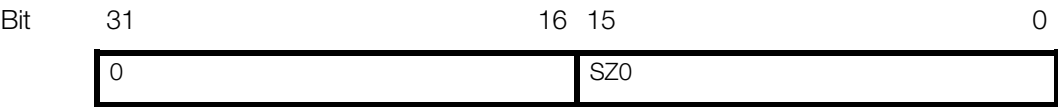
SXY0 = SXY1;

SXY1 = SXY2;

SXY2 = SXYP;

Register number: Data #16

Register name: SZ0
 Access: R/W
 Bit pattern:

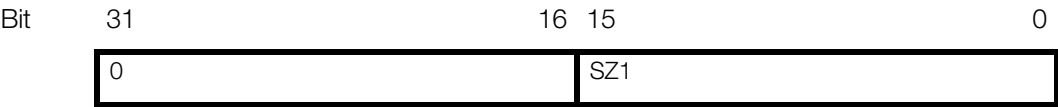


Fields:
 SZ0 (0.16.0) Screen coordinate Z-element. Note that this value was obtained in the calculation four times previous.

Internal operations:
 See Data #19: SZ3.

Register number: Data #17

Register name: SZ1
 Access: R/W
 Bit pattern:



Fields:
 SZ1 (0.16.0) Screen coordinate Z-element. Note that this value was obtained in the calculation three times previous.

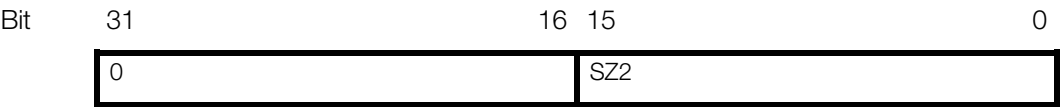
Internal operations:
 See Data #19: SZ3.

Register number: Data #18

Register name: SZ2

Access: R/W

Bit pattern:



Fields:

SZ2	(0.16.0)	Screen coordinate Z-element. Note that this value was obtained in the calculation two times previous.
-----	----------	---

Internal operations:

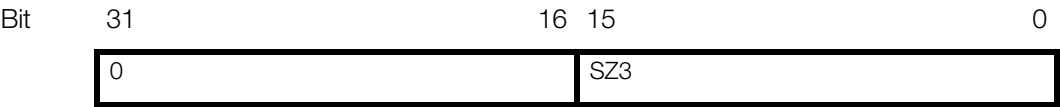
See Data #19: SZ3.

Register number: Data #19

Register name: SZ3

Access: R/W

Bit pattern:



Fields:

SZ3	(0.16.0)	Screen coordinate Z-element. Note that this value was obtained in the calculation one time previous.
-----	----------	--

Internal operations:

In several GTE instructions, substitutions are made in the following sequence.

SZ0 = SZ1;

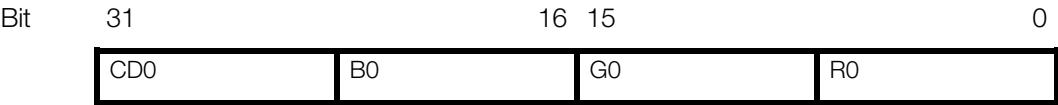
SZ1 = SZ2;

SZ2 = SZ3;

SZ3 = Coordinate Z-element obtained through calculation.

Register number: Data #20

Register name: RGB0
 Access: R/W
 Bit pattern:

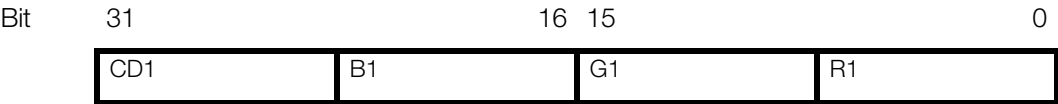


Fields:		
R0	(0.8.0)	Characteristic color R-element
G0	(0.8.0)	Characteristic color G-element
B0	(0.8.0)	Characteristic color B-element
CD0	(-.8.-)	Arbitrary 8-bit data

Internal operations:
 See Data #22: RGB2.

Register number: Data #21

Register name: RGB1
 Access: R/W
 Bit pattern:



Fields:		
R1	(0.8.0)	Characteristic color R-element
G1	(0.8.0)	Characteristic color G-element
B1	(0.8.0)	Characteristic color B-element
CD1	(-.8.-)	Arbitrary 8-bit data

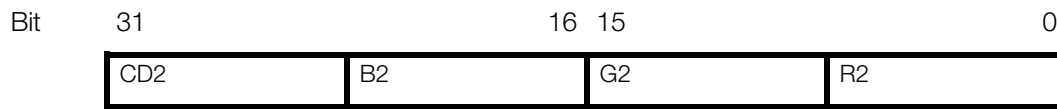
Internal operations:
 See Data #22: RGB2.

Register number: Data #22

Register name: RGB2

Access: R/W

Bit pattern:



Fields:

R2	(0.8.0)	Characteristic color R-element
----	---------	--------------------------------

G2	(0.8.0)	Characteristic color G-element
----	---------	--------------------------------

B2	(0.8.0)	Characteristic color B-element
----	---------	--------------------------------

CD2 (-.8.-) Arbitrary 8-bit data

Internal operations:

When several GTE instructions are executed, substitutions are made in the following sequence.

```
R0 = R1;
```

R1 = R2;

R2 = RGB Register R-field

G0 = G1;

G1 = G2;

G2 = RGB Register G-field

B0 = B1;

B1 = B2;

B2 = RGB Register B-field

```
CD0 = CD1;
```

CD1 = CD2;

CD2 = Bit pattern of GTE instruction currently being executed.

Register number: Data #23

Register name: RES1
Access: Prohibited

Register number: Data #24

Register name: MAC0
Access: R/W
Bit pattern:

Bit 31 0



Fields:
MAC0 (1.31.0) Sum of products value #0

Register number: Data #25

Register name: MAC1
Access: R/W
Bit pattern:

Bit 31 0



Fields:
MAC1 (1.31.0) Sum of products value #1

Register number: Data #26

Register name: MAC2

Access: R/W

Bit pattern:

Bit	31	0
-----	----	---



Fields:

MAC2 (1.31.0) Sum of products value #2

Register number: Data #27

Register name: MAC3

Access: R/W

Bit pattern:

Bit 31 0



Fields:

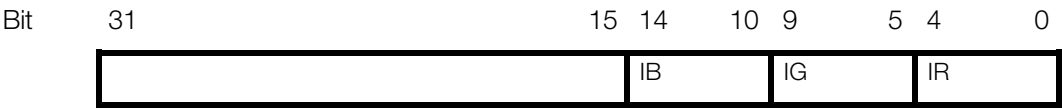
MAC3	(1.31.0)	Sum of products value #3
------	----------	--------------------------

Register number: Data #28

Register name: IRGB

Access: W

Bit pattern:



Fields:

IR	(-.5.-)	Color data (R-element) to be set as intermediate value
IG	(-.5.-)	Color data (G-element) to be set as intermediate value
IB	(-.5.-)	Color data (B-element) to be set as intermediate value

Internal operations:

The following processing is accomplished by writing data to this register.

IR1 = The value which format-converted R to (1.11.4)

IR2 = The value which format-converted G to (1.11.4)

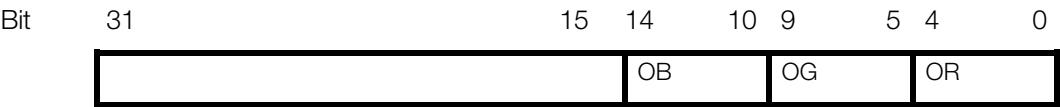
IR3 = The value which format-converted B to (1.11.4)

Register number: Data #29

Register name: ORGB

Access: R

Bit pattern:



Fields:

OR	(-.5.-)	Color data generated from intermediate value (R-element)
OG	(-.5.-)	Color data generated from intermediate value (G-element)
OB	(-.5.-)	Color data generated from intermediate value (B-element)

Internal operations:

By reading data from this register, the following operations are performed, including substitutions to each field.

OR = (IR1>>7)&0x1f;

OG = (IR2>>7)&0x1f;

OB = (IR3>>7)&0x1f;

The results obtained are then read.

Register number: Data #30

Register name: LZCS

Access: W

Bit pattern:

Bit 31 0



Fields:

LZCS	(1.31.0)	LZC source data
------	----------	-----------------

Internal operations:

See Data #31: LZCR.

Register number: Data #31

Register name: LZCR

Access: R

Bit pattern:

[illegible]

Fields:

LZCR	(0.6.0)	Leading zero/one count calculation result
------	---------	---

Internal operations:

By reading data from this register, the following operations are performed, including substitutions to each field.

Data #30: If the value of LZCS is positive,
LZCR = Leading zero count of LZCS value.

Data #30: If the value of LZCS is negative, LZCR = Leading one count of LZCS value. The results obtained are then read.

Chapter 6:

GTE Commands Reference

Limiters

During some calculation processing, calculation results, data values in registers, etc., are clipped when they exceed specified upper limit and lower limit values. In other words, data values lower than the lower limit value are converted to the lower limit value, and data values higher than the upper limit value are converted to the upper limit value. Also, the occurrence of such conversions is reflected in out-of-bounds data detection flags in the FLAG register. These functions are referred to as "limiters".

The usage of the various limiters and the codes used to specify them in this documentation are listed below.

Code Specification	Limiter	Out-of-bounds detect bit	Lower limit	Upper limit	Comments
A1S	A1	24	-2^{15}	$2^{15}-1$	
A2S	A2	23	-2^{15}	$2^{15}-1$	
A3S	A3	22	-2^{15}	$2^{15}-1$	
A1U	A1	24	0	$2^{15}-1$	
A2U	A2	23	0	$2^{15}-1$	
A3U	A3	22	0	$2^{15}-1$	
A1C	A1	24	0 or -2^{15}	$2^{15}-1$	Lower limit value is specified using lim argument.
A2C	A3	23	0 or -2^{15}	$2^{15}-1$	Lower limit value is specified using lim argument.
A3C	A3	22	0 or -2^{15}	$2^{15}-1$	Lower limit value is specified using lim argument.
B1	B1	21	0	2^8-1	
B2	B2	20	0	2^8-1	
B3	B3	19	0	2^8-1	
C	C	18	0	$2^{16}-1$	
D1	D1	14	-2^{10}	$2^{10}-1$	
D2	D2	13	-2^{10}	$2^{10}-1$	
E	E	12	0	$2^{12}-1$	

Calculation Error Detection

Overflow and underflow detection are performed only for certain specific calculation operations. In this documentation, the calculation test result flag number is listed between angle brackets < > to the right of calculation operations that are subject to such detection.

Explanations

Character attributes	Example	Content
Underline	<u>VAL</u>	Intermediate value (No corresponding register)
Embolden	OBJ	32-bit value

Code	Meaning
limX()	Limiter application X is the limiter designation code.
<-	FIFO data transfer Example: a <- b <- c <- d; is equivalent to the following: a = b; b = c; c = d;
^	Power Example: a^b means "a to the power of b".
==	Argument value condition
n=0,1,2 {}	Repeat the process between the curly parentheses {} three times, substituting 0, 1, and 2 for n.
(a.b.c)	Fixed-point number Sign portion: a bits, integer portion: b bits, fractional portion: c bits
(-b.-)	b-bit binary data
<n>	Calculation subject to overflow and underflow testing The test result is reflected in cumulative test flag n in the FLAG register.

Descriptor examples:

(A)

(1.15.0) A = B;

--> A=B is executed for the fixed-point expression (1.15.0).

(B)

(1.15.0) IR0 = limX(SSX);

Limiter:

Code	Lower limit	Upper limit
limX	-2¹⁵	2¹⁵-1

--> The 32-bit value SSX is rounded using the limiter specified by X. The fixed-point expression (1.15.0) representing the results obtained is substituted for IR1.

(C)

n=0,1,2{

(1.3.12)L1n
= limA(LL1n);

}

--> For the fixed-point expression (1.3.12), the following are executed:

L10=limA(LL10);

L11=limA(LL11);

L12=limA(LL12);

(D)

sf==0 sf==1

(1.31.0) (1.19.12) A = B;

--> B is substituted for A. However, the value is converted into a 32-bit signed fixed-point number with no fractional part if sf is 0, and with a 12-bit fraction if sf is 1.

Command List

Command	Required cycles	Function
RTPS	14	Coordinate transformation & perspective transformation
RTPT	22	Coordinate transformation & perspective transformation
NCDS	19	Light source calculation
NCDT	44	Light source calculation
NCCS	17	Light source calculation
NCCT	39	Light source calculation
CDP	13	Light source calculation
CC	11	Light source calculation
NCS	14	Light source calculation
NCT	30	Light source calculation
MVMVA	8	Matrix calculation
INTPL	8	Interpolation
DPCL	8	Depth queuing
DPCS	8	Depth queuing
DPCT	17	Depth queuing
SQR	5	Vector squaring
AVSZ3	5	Z-averaging
AVSZ4	6	Z-averaging
NCLIP	8	Normal clipping
OP	6	Outer product
GPF	5	General purpose interpolation
GPL	5	General purpose interpolation

Command Details

Command details are listed on the pages which follow.

RTPS Required cycles: 14**Function:** Coordinate transformation and perspective transformation**Calculations:**

- (1.31.12) $\underline{\mathbf{SSX}} = \mathbf{TRX} + R11*VX0 + R12*VY0 + R13*VZ0; <1>$
- (1.31.12) $\underline{\mathbf{SSY}} = \mathbf{TRY} + R21*VX0 + R22*VY0 + R23*VZ0; <2>$
- (1.31.12) $\underline{\mathbf{SSZ}} = \mathbf{TRZ} + R31*VX0 + R32*VY0 + R33*VZ0; <3>$
- (1.15. 0) $IR1 = \text{limA1S}(\underline{\mathbf{SSX}});$
- (1.15. 0) $IR2 = \text{limA2S}(\underline{\mathbf{SSY}});$
- (1.15. 0) $IR3 = \text{limA3S}(\underline{\mathbf{SSZ}});$
- (0.16. 0) $SZx(0) <- SZ0(1) <- SZ1(2) <- SZ2(3) <- \text{limC}(\underline{\mathbf{SSZ}});$
- (1.27.16) $\underline{\mathbf{SX}} = OFX + IR1*(H/SZ); <4>$
- (1.27.16) $\underline{\mathbf{SY}} = OFY + IR2*(H/SZ); <4>$
- (1.19.24) $\underline{\mathbf{P}} = DQB + DQA*(H/SZ); <4>$
- (1. 3.12) $IR0 = \text{limE}(\underline{\mathbf{P}})$
- (1.15. 0) $SX0 <- SX1 <- SX2 <- \text{limD1}(\underline{\mathbf{SX}});$
- (1.15. 0) $SY0 <- SY1 <- SY2 <- \text{limD2}(\underline{\mathbf{SY}});$
- (1. 7.24) $\mathbf{MAC0} = \underline{\mathbf{P}};$
- (1.31. 0) $\mathbf{MAC1} = \underline{\mathbf{SSX}};$
- (1.31. 0) $\mathbf{MAC2} = \underline{\mathbf{SSY}};$
- (1.31. 0) $\mathbf{MAC3} = \underline{\mathbf{SSZ}};$

RTPS**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

RTPT Required cycles: 22**Function:** Coordinate transformation and perspective transformation**Calculations:**

n=0,1,2 {

$$(1.31.12) \quad \underline{\mathbf{SSXn}} = \mathbf{TRX} + R11*VXn + R12*VYn + R13*VZn; <1>$$

$$(1.31.12) \quad \underline{\mathbf{SSYn}} = \mathbf{TRY} + R21*VXn + R22*VYn + R23*VZn; <2>$$

$$(1.31.12) \quad \underline{\mathbf{SSZn}} = \mathbf{TRZ} + R31*VXn + R32*VYn + R33*VZn; <3>$$

$$(0.16.0) \quad SZx(0) = SZ2(3);$$

$$(0.16.0) \quad SZ0(1) = \text{limC}(\underline{\mathbf{SSZ0}});$$

$$(0.16.0) \quad SZ0(2) = \text{limC}(\underline{\mathbf{SSZ1}});$$

$$(0.16.0) \quad SZ0(3) = \text{limC}(\underline{\mathbf{SSZ2}});$$

$$(1.27.16) \quad \underline{\mathbf{SXn}} = \mathbf{OFX} + \mathbf{IR1}*(\mathbf{H}/\mathbf{SZn}); <4>$$

$$(1.27.16) \quad \underline{\mathbf{SYn}} = \mathbf{OFY} + \mathbf{IR2}*(\mathbf{H}/\mathbf{SZn}); <4>$$

$$(1.19.24) \quad \underline{\mathbf{P}} = \mathbf{DQB} + \mathbf{DQA}*(\mathbf{H}/\mathbf{SZ2}); <4>$$

$$(1.3.12) \quad \mathbf{IR0} = \text{limE}(\underline{\mathbf{P}});$$

$$(1.15.0) \quad \mathbf{SXn} = \text{limD1}(\underline{\mathbf{SXn}});$$

$$(1.15.0) \quad \mathbf{SYn} = \text{limD2}(\underline{\mathbf{SYn}});$$

}

$$(1.15.0) \quad \mathbf{IR1} = \text{limA1S}(\underline{\mathbf{SSX2}});$$

$$(1.15.0) \quad \mathbf{IR2} = \text{limA2S}(\underline{\mathbf{SSY2}});$$

$$(1.15.0) \quad \mathbf{IR3} = \text{limA3S}(\underline{\mathbf{SSZ2}});$$

$$(1.7.24) \quad \mathbf{MAC0} = \underline{\mathbf{P}};$$

$$(1.31.0) \quad \mathbf{MAC1} = \underline{\mathbf{SSX2}};$$

$$(1.31.0) \quad \mathbf{MAC2} = \underline{\mathbf{SSY2}};$$

$$(1.31.0) \quad \mathbf{MAC3} = \underline{\mathbf{SSZ2}};$$

RTPT**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

NCDS Required cycles: 19**Function:** Light source calculation**Calculations:**

(1.19.24) $\underline{\mathbf{LL1}} = L11*VX0 + L12*VY0 + L13*VZ0; <1>$

(1.19.24) $\underline{\mathbf{LL2}} = L21*VX0 + L22*VY0 + L23*VZ0; <2>$

(1.19.24) $\underline{\mathbf{LL3}} = L31*VX0 + L32*VY0 + L33*VZ0; <3>$

(1. 3.12) $L1 = \text{limA1U}(\underline{\mathbf{LL1}});$

(1. 3.12) $L2 = \text{limA2U}(\underline{\mathbf{LL2}});$

(1. 3.12) $L3 = \text{limA3U}(\underline{\mathbf{LL3}});$

(1.19.24) $\underline{\mathbf{RRLT}} = \mathbf{RBK} + LR1*L1 + LR2*L2 + LR3*L3; <1>$

(1.19.24) $\underline{\mathbf{GGLT}} = \mathbf{GBK} + LG1*L1 + LG2*L2 + LG3*L3; <2>$

(1.19.24) $\underline{\mathbf{BBLT}} = \mathbf{BBK} + LB1*L1 + LB2*L2 + LB3*L3; <3>$

(1. 3.12) $RLT = \text{limA1U}(\underline{\mathbf{RRLT}});$

(1. 3.12) $GLT = \text{limA2U}(\underline{\mathbf{GGLT}});$

(1. 3.12) $BLT = \text{limA3U}(\underline{\mathbf{BBLT}});$

(1.27.16) $\underline{\mathbf{RR0}} = R*RLT + IR0*\text{limA1S}(\mathbf{RFC} - R*RLT); <1>$

(1.27.16) $\underline{\mathbf{GG0}} = G*GLT + IR0*\text{limA2S}(\mathbf{GFC} - G*GLT); <2>$

(1.27.16) $\underline{\mathbf{BB0}} = B*BLT + IR0*\text{limA3S}(\mathbf{BFC} - B*BLT); <3>$

(1.11. 4) $IR1 = \text{limA1U}(\underline{\mathbf{RR0}});$

(1.11. 4) $IR2 = \text{limA2U}(\underline{\mathbf{GG0}});$

(1.11. 4) $IR3 = \text{limA3U}(\underline{\mathbf{BB0}});$

(-. 8. -) $CD0 <- CD1 <- CD2 <- CODE$

(0. 8. 0) $R0 <- R1 <- R2 <- \text{limB1}(\underline{\mathbf{RR0}});$

(0. 8. 0) $G0 <- G1 <- G2 <- \text{limB2}(\underline{\mathbf{GG0}});$

(0. 8. 0) $B0 <- B1 <- B2 <- \text{limB3}(\underline{\mathbf{BB0}});$

(1.27. 4) $\mathbf{MAC1} = \underline{\mathbf{RR0}};$

(1.27. 4) $\mathbf{MAC2} = \underline{\mathbf{GG0}};$

(1.27. 4) $\mathbf{MAC3} = \underline{\mathbf{BB0}};$

NCDS**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

NCDT Required cycles: 44**Function:** Light source calculation**Calculations:**

n=0,1,2 {

$$(1.19.24) \quad \underline{\mathbf{LL1n}} = L11*VXn + L12*Vyn + L13*VZn; <1>$$

$$(1.19.24) \quad \underline{\mathbf{LL2n}} = L21*VXn + L22*Vyn + L23*VZn; <2>$$

$$(1.19.24) \quad \underline{\mathbf{LL3n}} = L31*VXn + L32*Vyn + L33*VZn; <3>$$

$$(1.3.12) \quad L1n = \text{limA1U}(\underline{\mathbf{LL1n}});$$

$$(1.3.12) \quad L2n = \text{limA2U}(\underline{\mathbf{LL2n}});$$

$$(1.3.12) \quad L3n = \text{limA3U}(\underline{\mathbf{LL3n}});$$

$$(1.19.24) \quad \underline{\mathbf{RRLTn}} = \mathbf{RBK} + LR1*L1n + LR2*L2n + LR3*L3n; <1>$$

$$(1.19.24) \quad \underline{\mathbf{GGLTn}} = \mathbf{GBK} + LG1*L1n + LG2*L2n + LG3*L3n; <2>$$

$$(1.19.24) \quad \underline{\mathbf{BBLTn}} = \mathbf{BBK} + LB1*L1n + LB2*L2n + LB3*L3n; <3>$$

$$(1.3.12) \quad \mathbf{RLTn} = \text{limA1U}(\underline{\mathbf{RRLTn}});$$

$$(1.3.12) \quad \mathbf{GLTn} = \text{limA2U}(\underline{\mathbf{GGLTn}});$$

$$(1.3.12) \quad \mathbf{BLTn} = \text{limA3U}(\underline{\mathbf{BBLTn}});$$

$$(1.27.16) \quad \underline{\mathbf{RRn}} = R*\mathbf{RLTn} + IR0*\text{limA1S}(\mathbf{RFC} - R*\mathbf{RLTn}); <1>$$

$$(1.27.16) \quad \underline{\mathbf{GGn}} = G*\mathbf{GLTn} + IR0*\text{limA2S}(\mathbf{GFC} - G*\mathbf{GLTn}); <2>$$

$$(1.27.16) \quad \underline{\mathbf{BBn}} = B*\mathbf{BLTn} + IR0*\text{limA3S}(\mathbf{BFC} - B*\mathbf{BLTn}); <3>$$

$$(-.8.-) \quad \mathbf{CDn} = \mathbf{CODE}$$

$$(-.8.0) \quad \mathbf{Rn} = \text{limB1}(\underline{\mathbf{RRn}}); \mathbf{Gn} = \text{limB2}(\underline{\mathbf{GGn}});$$

$$(-.8.0) \quad \mathbf{Bn} = \text{limB3}(\underline{\mathbf{BBn}});$$

}

$$(1.11.4) \quad \mathbf{IR1} = \text{limA1U}(\underline{\mathbf{RR2}});$$

$$(1.11.4) \quad \mathbf{IR2} = \text{limA2U}(\underline{\mathbf{GG2}});$$

$$(1.11.4) \quad \mathbf{IR3} = \text{limA3U}(\underline{\mathbf{BB2}});$$

$$(1.27.4) \quad \mathbf{MAC1} = \underline{\mathbf{RR2}};$$

$$(1.27.4) \quad \mathbf{MAC2} = \underline{\mathbf{GG2}};$$

$$(1.27.4) \quad \mathbf{MAC3} = \underline{\mathbf{BB2}};$$

NCDT**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

NCCS Required cycles: 17**Function:** Light source calculation**Calculations:**

(1.19.24) **LL1** = L11*VX0 + L12*VY0 + L13*VZ0; <1>
 (1.19.24) **LL2** = L21*VX0 + L22*VY0 + L23*VZ0; <2>
 (1.19.24) **LL3** = L31*VX0 + L32*VY0 + L33*VZ0; <3>
 (1. 3.12) L1 = limA(**LL1**);
 (1. 3.12) L2 = limA(**LL2**);
 (1. 3.12) L3 = limA(**LL3**);
 (1.19.24) **RRLT** = RBK + LR1*L1 +LR2*L2 + LR3*L3; <1>
 (1.19.24) **GGLT** = GBK + LG1*L1 +LG2*L2 + LG3*L3; <2>
 (1.19.24) **BBLT** = BBK + LB1*L1 +LB2*L2 + LB3*L3; <3>
 (1. 3.12) RLT = limA1U(**RRLT**);
 (1. 3.12) GLT = limA2U(**GGLT**);
 (1. 3.12) BLT = limA3U(**BBLT**);
 (1.27.16) **RR0** = R*RLT; <1>
 (1.27.16) **GG0** = G*GLT; <2>
 (1.27.16) **BB0** = B*BLT; <3>
 (1.11. 4) IR1 = limA1U(**RR0**);
 (1.11. 4) IR2 = limA2U(**GG0**);
 (1.11. 4) IR3 = limA3U(**BB0**);
 (-. 8. -) CD0 <- CD1 <- CD2 <- CODE
 (0. 8. 0) R0 <- R1 <- R2 <- limB1(**RR0**);
 (0. 8. 0) G0 <- G1 <- G2 <- limB2(**GG0**);
 (0. 8. 0) B0 <- B1 <- B2 <- limB3(**BB0**);
 (1.27. 4) **MAC1** = **RR0**;
 (1.27. 4) **MAC2** = **GG0**;
 (1.27. 4) **MAC3** = **BB0**;

NCCS**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

NCCT Required cycles: 39**Function:** Light source calculation**Calculations:**

n=0,1,2 {

$$(1.19.24) \quad \underline{LL1n} = L11*VXn + L12*VYn + L13*VZn; <1>$$

$$(1.19.24) \quad \underline{LL2n} = L21*VXn + L22*VYn + L23*VZn; <2>$$

$$(1.19.24) \quad \underline{LL3n} = L31*VXn + L32*VYn + L33*VZn; <3>$$

$$(1.3.12) \quad L1n = \text{limA1U}(\underline{LL1n});$$

$$(1.3.12) \quad L2n = \text{limA2U}(\underline{LL2n});$$

$$(1.3.12) \quad L3n = \text{limA3U}(\underline{LL3n});$$

$$(1.19.24) \quad \underline{RRLTn} = \underline{RBK} + LR1*L1n + LR2*L2n + LR3*L3n; <1>$$

$$(1.19.24) \quad \underline{GGLTn} = \underline{GBK} + LG1*L1n + LG2*L2n + LG3*L3n; <2>$$

$$(1.19.24) \quad \underline{BBLTn} = \underline{BBK} + LB1*L1n + LB2*L2n + LB3*L3n; <3>$$

$$(1.3.12) \quad RLTn = \text{limA1U}(\underline{RRLTn});$$

$$(1.3.12) \quad GLTn = \text{limA2U}(\underline{GGLTn});$$

$$(1.3.12) \quad BLTn = \text{limA3U}(\underline{BBLTn});$$

$$(1.27.16) \quad \underline{RRn} = R*RLTn; <1>$$

$$(1.27.16) \quad \underline{GGn} = G*GLTn; <2>$$

$$(1.27.16) \quad \underline{BBn} = B*BLTn; <3>$$

$$(-.8.-) \quad CDn = \text{CODE}$$

$$(0.8.0) \quad Rn = \text{limB1}(\underline{RRn});$$

$$(0.8.0) \quad Gn = \text{limB2}(\underline{GGn});$$

$$(0.8.0) \quad Bn = \text{limB3}(\underline{BBn});$$

}

$$(1.11.4) \quad IR1 = \text{limA1U}(\underline{RR2}); IR2 = \text{limA2U}(\underline{GG2});$$

$$(1.11.4) \quad IR3 = \text{limA3U}(\underline{BB2});$$

$$(1.27.4) \quad \underline{MAC1} = \underline{RR2};$$

$$(1.27.4) \quad \underline{MAC2} = \underline{GG2};$$

$$(1.27.4) \quad \underline{MAC3} = \underline{BB2};$$

NCCT**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

CDP Required cycles: 13**Function:** Light source calculation**Calculations:**

(1.19.24) $\mathbf{RRLT} = \mathbf{RBK} + \mathbf{LR1} * \mathbf{IR1} + \mathbf{LR2} * \mathbf{IR2} + \mathbf{LR3} * \mathbf{IR3}; <1>$

(1.19.24) $\mathbf{GGLT} = \mathbf{GBK} + \mathbf{LG1} * \mathbf{IR1} + \mathbf{LG2} * \mathbf{IR2} + \mathbf{LG3} * \mathbf{IR3}; <2>$

(1.19.24) $\mathbf{BBLT} = \mathbf{BBK} + \mathbf{LB1} * \mathbf{IR1} + \mathbf{LB2} * \mathbf{IR2} + \mathbf{LB3} * \mathbf{IR3}; <3>$

(1. 3.12) $\mathbf{RLT} = \lim \mathbf{A1U}(\mathbf{RRLT});$

(1. 3.12) $\mathbf{GLT} = \lim \mathbf{A2U}(\mathbf{GGLT});$

(1. 3.12) $\mathbf{BLT} = \lim \mathbf{A3U}(\mathbf{BBLT});$

(1.27.16) $\mathbf{RR0} = \mathbf{R} * \mathbf{RLT} + \mathbf{IR0} * \lim \mathbf{A1S}(\mathbf{RFC} - \mathbf{R} * \mathbf{RLT}); <1>$

(1.27.16) $\mathbf{GG0} = \mathbf{G} * \mathbf{GLT} + \mathbf{IR0} * \lim \mathbf{A2S}(\mathbf{GFC} - \mathbf{G} * \mathbf{GLT}); <2>$

(1.27.16) $\mathbf{BB0} = \mathbf{B} * \mathbf{BLT} + \mathbf{IR0} * \lim \mathbf{A3S}(\mathbf{BFC} - \mathbf{B} * \mathbf{BLT}); <3>$

(1.11. 4) $\mathbf{IR1} = \lim \mathbf{A1U}(\mathbf{RR0});$

(1.11. 4) $\mathbf{IR2} = \lim \mathbf{A2U}(\mathbf{GG0});$

(1.11. 4) $\mathbf{IR3} = \lim \mathbf{A3U}(\mathbf{BB0});$

(-. 8. -) $\mathbf{CD0} <- \mathbf{CD1} <- \mathbf{CD2} <- \mathbf{CODE}$

(0. 8. 0) $\mathbf{R0} <- \mathbf{R1} <- \mathbf{R2} <- \lim \mathbf{B1}(\mathbf{RR0});$

(0. 8. 0) $\mathbf{G0} <- \mathbf{G1} <- \mathbf{G2} <- \lim \mathbf{B2}(\mathbf{GG0});$

(0. 8. 0) $\mathbf{B0} <- \mathbf{B1} <- \mathbf{B2} <- \lim \mathbf{B3}(\mathbf{BB0});$

(1.27. 4) $\mathbf{MAC1} = \mathbf{RR0};$

(1.27. 4) $\mathbf{MAC2} = \mathbf{GG0};$

(1.27. 4) $\mathbf{MAC3} = \mathbf{BB0};$

CDP**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

CC Required cycles: 11**Function:** Light source calculation**Calculations:**

(1.19.24) **RRLT** = **RBK** + LR1*IR1 + LR2*IR2 + LR3*IR3; <1>
 (1.19.24) **GGLT** = **GBK** + LG1*IR1 + LG2*IR2 + LG3*IR3; <2>
 (1.19.24) **BBLT** = **BBK** + LB1*IR1 + LB2*IR2 + LB3*IR3; <3>
 (1. 3.12) RLT = limA1U(**RRLT**);
 (1. 3.12) GLT = limA2U(**GGLT**);
 (1. 3.12) BLT = limA3U(**BBLT**);
 (1.27.16) **RR0** = R*RLT; <1>
 (1.27.16) **GG0** = G*GLT; <2>
 (1.27.16) **BB0** = B*BLT; <3>
 (1.11. 4) IR1 = limA1U(**RR0**);
 (1.11. 4) IR2 = limA2U(**GG0**);
 (1.11. 4) IR3 = limA3U(**BB0**);
 (-. 8. -) CD0 <- CD1 <- CD2 <- CODE
 (0. 8. 0) R0 <- R1 <- R2 <- limB1(**RR0**);
 (0. 8. 0) G0 <- G1 <- G2 <- limB2(**GG0**);
 (0. 8. 0) B0 <- B1 <- B2 <- limB3(**BB0**);
 (1.27. 4) **MAC1** = **RR0**;
 (1.27. 4) **MAC2** = **GG0**;
 (1.27. 4) **MAC3** = **BB0**;

CC**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

NCS Required cycles: 14**Function:** Light source calculation**Calculations:**

(1.19.24) **LL1** = L11*VX0 + L12*VY0 + L13*VZ0; <1>
 (1.19.24) **LL2** = L21*VX0 + L22*VY0 + L23*VZ0; <2>
 (1.19.24) **LL3** = L31*VX0 + L32*VY0 + L33*VZ0; <3>
 (1. 3.12) L1 = limA1U(**LL1**);
 (1. 3.12) L2 = limA2U(**LL2**);
 (1. 3.12) L3 = limA3U(**LL3**);
 (1.19.24) **RR0** = **RBK** + LR1*L1 + LR2*L2 + LR3*L3; <1>
 (1.19.24) **GG0** = **GBK** + LG1*L1 + LG2*L2 + LG3*L3; <2>
 (1.19.24) **BB0** = **BBK** + LB1*L1 + LB2*L2 + LB3*L3; <3>
 (1. 3.12) IR1 = limA1U(**RR0**);
 (1. 3.12) IR2 = limA2U(**GG0**);
 (1. 3.12) IR3 = limA3U(**BB0**);
 (-. 8. -) CD0 <- CD1 <- CD2 <- CODE
 (0. 0. 8) R0 <- R1 <- R2 <- limB1(**RR0**);
 (0. 0. 8) G0 <- G1 <- G2 <- limB2(**GG0**);
 (0. 0. 8) B0 <- B1 <- B2 <- limB3(**BB0**);
 (1.19.12) **MAC1** = **RR0**;
 (1.19.12) **MAC2** = **GG0**;
 (1.19.12) **MAC3** = **BB0**;

NCS**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

NCT Required cycles: 30**Function:** Light source calculation**Calculations:**

n=0,1,2 {

$$(1.19.24) \quad \underline{\mathbf{LL1n}} = L11*VXn + L12*VYn + L13*VZn; <1>$$

$$(1.19.24) \quad \underline{\mathbf{LL2n}} = L21*VXn + L22*VYn + L23*VZn; <2>$$

$$(1.19.24) \quad \underline{\mathbf{LL3n}} = L31*VXn + L32*VYn + L33*VZn; <3>$$

$$(1.3.12) \quad L1n = \text{limA1U}(\underline{\mathbf{LL1n}});$$

$$(1.3.12) \quad L2n = \text{limA2U}(\underline{\mathbf{LL2n}});$$

$$(1.3.12) \quad L3n = \text{limA3U}(\underline{\mathbf{LL3n}});$$

$$(1.19.24) \quad \underline{\mathbf{RRn}} = \mathbf{RBK} + LR1*L1n + LR2*L2n + LR3*L3n; <1>$$

$$(1.19.24) \quad \underline{\mathbf{GGn}} = \mathbf{GBK} + LG1*L1n + LG2*L2n + LG3*L3n; <2>$$

$$(1.19.24) \quad \underline{\mathbf{BBn}} = \mathbf{BBK} + LB1*L1n + LB2*L2n + LB3*L3n; <3>$$

$$(-.8.-) \quad CDn = \text{CODE}$$

$$(0.0.8) \quad Rn = \text{limB1}(\underline{\mathbf{RRn}});$$

$$(0.0.8) \quad Gn = \text{limB2}(\underline{\mathbf{GGn}});$$

$$(0.0.8) \quad Bn = \text{limB3}(\underline{\mathbf{BBn}});$$

}

$$(1.3.12) \quad IR1 = \text{limA1U}(\underline{\mathbf{RR2}});$$

$$(1.3.12) \quad IR2 = \text{limA2U}(\underline{\mathbf{GG2}});$$

$$(1.3.12) \quad IR3 = \text{limA3U}(\underline{\mathbf{BB2}});$$

$$(1.19.12) \quad \mathbf{MAC1} = \underline{\mathbf{RR2}};$$

$$(1.19.12) \quad \mathbf{MAC2} = \underline{\mathbf{GG2}};$$

$$(1.19.12) \quad \mathbf{MAC3} = \underline{\mathbf{BB2}};$$

NCT**Referenced registers:**

	Data		Control
0	VX0,VY0		R11,R12
1	VZ0		R13,R21
2	VX1,VY1		R22,R23
3	VZ1		R31,R32
4	VX2,VY2		R33
5	VZ2		TRX
6	RGB	CODE	TRY
7	OTZ		TRZ
8	IR0		L11,L12
9	IR1		L13,L21
10	IR2		L22,L23
11	IR3		L31,L32
12	SX0,SY0		L33
13	SX1,SY1		RBK
14	SX2,SY2		GBK
15	SX2P,SY2P		BBK
16	SZx(0)		LR1,LR2
17	SZ0(1)		LR3,LG1
18	SZ1(2)		LG2,LG3
19	SZ2(3)		LB1,LB2
20	R0 G0 B0 CD0		LB3
21	R1 G1 B1 CD1		RFC
22	R2 G2 B2 CD2		GFC
23			BFC
24	MAC0		OFX
25	MAC1		OFY
26	MAC2		H
27	MAC3		DQA
28	IRGB		DQB
29	ORGB		ZSF3
30	DATA32		ZSF4
31	LZC		FLAG

Modified registers:

	Data		Control
0	VX0,VY0		R11,R12
1	VZ0		R13,R21
2	VX1,VY1		R22,R23
3	VZ1		R31,R32
4	VX2,VY2		R33
5	VZ2		TRX
6	RGB	CODE	TRY
7	OTZ		TRZ
8	IR0		L11,L12
9	IR1		L13,L21
10	IR2		L22,L23
11	IR3		L31,L32
12	SX0,SY0		L33
13	SX1,SY1		RBK
14	SX2,SY2		GBK
15	SX2P,SY2P		BBK
16	SZx(0)		LR1,LR2
17	SZ0(1)		LR3,LG1
18	SZ1(2)		LG2,LG3
19	SZ2(3)		LB1,LB2
20	R0 G0 B0 CD0		LB3
21	R1 G1 B1 CD1		RFC
22	R2 G2 B2 CD2		GFC
23			BFC
24	MAC0		OFX
25	MAC1		OFY
26	MAC2		H
27	MAC3		DQA
28	IRGB		DQB
29	ORGB		ZSF3
30	DATA32		ZSF4
31	LZC		FLAG

MVMVA sf,mx,v,cv,lm Required cycles: 8**Function:** Matrix and vector multiplication**Items specified using arguments:**

Argument	Specified content	Value=0	Value = 1	Value = 2	Value = 3
sf	Scaling format	Scale large	Scale small	Not valid	Not valid
mx	Multiplication array (MX) (1.3.12)	R	L	LR	Not valid
v	Multiplication vector (V) (1.m.n)	Vp0 p=X/Y/Z	Vp1 p=X/Y/Z	Vp2 p=X/Y/Z	IRp p=0/1/2
cv	Addition vector (CV) (1.16+m. n)	TRp p=X/Y/Z	pBK p=R/B/G	pFC p=R/B/G	0
lm	Limiter A1/2/3 lower limit	-2^15	0	Not valid	Not valid

*** Data formats**

The multiplication matrix data format is fixed.

The other data formats are determined by the multiplication vector data format.

Calculations: (m and n are determined by the multiplication vector data format.)(1.16+m.n+12) **MT1** = CV1 + MX11*V1 + MX12*V2 + MX13*V3; <1>(1.16+m.n+12) **MT2** = CV2 + MX21*V1 + MX22*V2 + MX23*V3; <2>(1.16+m.n+12) **MT3** = CV3 + MX31*V1 + MX32*V2 + MX33*V3; <3>(1.16+m.n) **MAC1** = **MT1**;(1.16+m.n) **MAC2** = **MT2**;(1.16+m.n) **MAC3** = **MT3**;

sf == 0 sf == 1

(1.m-12.n+12) (1.m.n) IR1 = limA1C(**MT1**);(1.m-12.n+12) (1.m.n) IR2 = limA2C(**MT2**);(1.m-12.n+12) (1.m.n) IR3 = limA3C(**MT3**);

MVMVA sf,mx,v,cv,lm**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

INTPL Required cycles: 8**Function:** Interpolation**Calculations:** (m and n specify the data format of IRp(p=1,2,3) as (1.m.n).)(1.16+m.n+12) **IRL1** = 1.0*IR1 + IR0*limA1S(**RFC**-1.0*IR1); <1>(1.16+m.n+12) **IRL2** = 1.0*IR2 + IR0*limA2S(**GFC**-1.0*IR2); <2>(1.16+m.n+12) **IRL3** = 1.0*IR3 + IR0*limA3S(**BFC**-1.0*IR3); <3>(1.m.n) IR1 = limA1S(**IPL1**);(1.m.n) IR2 = limA2S(**IPL2**);(1.m.n) IR3 = limA3S(**IPL3**);

(-.8.-) CD0 <- CD1 <- CD2 <- CODE

(0.12-n.n-4) R0 <- R1 <- R2 <- limB1(**IPL1**);(0.12-n.n-4) G0 <- G1 <- G2 <- limB2(**IPL2**);(0.12-n.n-4) B0 <- B1 <- B2 <- limB3(**IPL3**);(1.16+m.n) **MAC1** = **IPL1**;(1.16+m.n) **MAC2** = **IPL2**;(1.16+m.n) **MAC3** = **IPL3**;

INTPL**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

DPCL Required cycles: 8**Function:** Depth queuing**Calculations:**

- (1.27.16) $\underline{RR0} = R * IR1 + IR0 * \text{limA1S}(\underline{RFC} - R * IR1); <1>$
- (1.27.16) $\underline{GG0} = G * IR2 + IR0 * \text{limA2S}(\underline{GFC} - G * IR2); <2>$
- (1.27.16) $\underline{BB0} = B * IR3 + IR0 * \text{limA3S}(\underline{BFC} - B * IR3); <3>$
- (1.11. 4) $IR1 = \text{limA1S}(\underline{RR0});$
- (1.11. 4) $IR2 = \text{limA2S}(\underline{GG0});$
- (1.11. 4) $IR3 = \text{limA3S}(\underline{BB0});$
- (-. 8. -) $CD0 <- CD1 <- CD2 <- CODE$
- (0. 8. 0) $R0 <- R1 <- R2 <- \text{limB1}(\underline{RR0});$
- (0. 8. 0) $G0 <- G1 <- G2 <- \text{limB2}(\underline{GG0});$
- (0. 8. 0) $B0 <- B1 <- B2 <- \text{limB3}(\underline{BB0});$
- (1.27. 4) $\underline{MAC1} = \underline{RR0};$
- (1.27. 4) $\underline{MAC2} = \underline{GG0};$
- (1.27. 4) $\underline{MAC3} = \underline{BB0};$

DPCL**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

DPCS Required cycles: 8**Function:** Depth queuing**Calculations:**

(1.27.16) $\underline{RR0} = R*1.0 + IR0*limA1S(\underline{RFC}-R*1.0); <1>$

(1.27.16) $\underline{GG0} = G*1.0 + IR0*limA2S(\underline{GFC}-G*1.0); <2>$

(1.27.16) $\underline{BB0} = B*1.0 + IR0*limA3S(\underline{BFC}-B*1.0); <3>$

(1.11. 4) $IR1 = limA1S(\underline{RR0});$

(1.11. 4) $IR2 = limA2S(\underline{GG0});$

(1.11. 4) $IR3 = limA3S(\underline{BB0});$

(-.8.-) $CD0 <- CD1 <- CD2 <- CODE$

(0. 8. 0) $R0 <- R1 <- R2 <- limB1(\underline{RR0});$

(0. 8. 0) $G0 <- G1 <- G2 <- limB2(\underline{GG0});$

(0. 8. 0) $B0 <- B1 <- B2 <- limB3(\underline{BB0});$

(1.27. 4) $\underline{MAC1} = \underline{RR0};$

(1.27. 4) $\underline{MAC2} = \underline{GG0};$

(1.27. 4) $\underline{MAC3} = \underline{BB0};$

DPCS**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

DPCT Required cycles: 17**Function:** Depth queuing**Calculations:**

```

n=0,1,2 {
  (1.27.16)   RRn = Rn*1.0 + IR0*limA1S(RFC-R*1.0); <1>
  (1.27.16)   G Gn = Gn*1.0 + IR0*limA2S(GFC-G*1.0); <2>
  (1.27.16)   BBn = Bn*1.0 + IR0*limA3S(BFC-B*1.0); <3>
  (1.11. 4)   IR1 = limA1S(RR2);
  (1.11. 4)   IR2 = limA2S(GG2);
  (1.11. 4)   IR3 = limA3S(BB2);
  (-.8.-)     CDn = CODE
  (0. 8. 0)   Rn = limB1(RRn);
  (0. 8. 0)   Gn = limB2(G Gn);
  (0. 8. 0)   Bn = limB3(BBn);
}
  (1.27. 4)   MAC1 = RR2;
  (1.27. 4)   MAC2 = GG2;
  (1.27. 4)   MAC3 = BB2;

```

DPCT**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

SQR sf
Required cycles: 5

Function: Vector squaring

Items specified using arguments:

Argument	Specified content	Value=0	Value=1
sf	Output format	--	Performs calculations on data shifted 12 bits to the left in the IRn register.

Calculations: (m and n specify the data format of IRp(p=1,2,3) as (1.m.n).)

sf == 0		sf == 1	
(1.m+28.n)	(1.m+16.n+12)	L1	= IR1*IR1; <1>
(1.m+28.n)	(1.m+16.n+12)	L2	= IR2*IR2; <2>
(1.m+28.n)	(1.m+16.n+12)	L3	= IR3*IR3; <3>
(1.m.n)	(1.m+16.n+12)	IR1 = limA1U(L1);	
(1.m.n)	(1.m+16.n+12)	IR2 = limA2U(L2);	
(1.m.n)	(1.m+16.n+12)	IR3 = limA3U(L3);	
(1.m+16.0)	(1.m+16.n+12)	MAC1 = L1 ;	
(1.m+16.0)	(1.m+16.n+12)	MAC2 = L2 ;	
(1.m+16.0)	(1.m+16.n+12)	MAC3 = L3 ;	

SQR sf**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

AVSZ3 **Required cycles: 5****Function:** Z-averaging**Calculations:**

(1.31.21) **OOTZ** = ZSF3*SZ0(1)
 + ZSF3*SZ1(2)
 + ZSF3*SZ2(3); <4>

(0.16. 0) OTZ = limC(**OOTZ**);

(1.31. 0) **MAC0** = **OOTZ**;

AVSZ3**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

AVSZ4 **Required cycles: 6****Function:** Z-averaging**Calculations:**

(1.31.12) **OOTZ** = ZSF4*SZx(0)
 + ZSF4*SZ0(1)
 + ZSF4*SZ1(2)
 + ZSF4*SZ2(3); <4>

(0.16. 0) OTZ = limC(**OOTZ**);

(1.31. 0) **MAC0** = **OOTZ**;

AVSZ4**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

NCLIP Required cycles: 8**Function:** Normal clipping**Calculations:**

$$(1.43. 0) \quad \underline{\mathbf{OPZ}} = \mathbf{SX0*SY1} + \mathbf{SX1*SY2} + \mathbf{SX2*SY0} \\ - \mathbf{SX0*SY2} - \mathbf{SX1*SY0} - \mathbf{SX2*SY1}; <4>$$

$$(1.31. 0) \quad \mathbf{MAC0} = \underline{\mathbf{OPZ}};$$

NCLIP**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

OP sf Required cycles: 6**Function:** Outer product**Items specified using arguments:**

Argument	Specified content	Value=0	Value=1
sf	Output format	--	Performs calculations on data shifted 12 bits to the left in the IRn register.

Calculations: (m and n specify the data format of IRp(p=1,2,3) as (1.m.n).)

sf == 0

sf == 1

(1.m+28.n) (1.m+16.n+12) **OPX** = DY1(R22)*DZ2(IR3)
 - DZ1(R33)*DY2(IR2); <1>

(1.m+28.n) (1.m+16.n+12) **OPY** = DZ1(R33)*DX2(IR1)
 - DX1(R11)*DZ2(IR3); <2>

(1.m+28.n) (1.m+16.n+12) **OPZ** = DX1(R11)*DY2(IR2)
 - DY1(R22)*DX2(IR1); <3>

(1.m.n) (1.m.n) IR1 = limA1S(**OPX**);

(1.m.n) (1.m.n) IR2 = limA2S(**OPY**);

(1.m.n) (1.m.n) IR3 = limA3S(**OPZ**);

(1.m+16.n) (1.m+16.n) **MAC1** = **OPX**;

(1.m+16.n) (1.m+16.n) **MAC2** = **OPY**;

(1.m+16.n) (1.m+16.n) **MAC3** = **OPZ**;

OP sf**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

GPF sf
Required cycles: 5

Function: General purpose interpolation

Items specified using arguments:

Argument	Specified content	Value=0	Value=1
sf	Output format	--	Performs calculations on data shifted 12 bits to the left in the IRn register.

Calculations: (m and n specify the data format of IRp(p=1,2,3) as (1.m.n).)

sf == 0

sf == 1

(1.m+28.n)

(1.m+16.n+12)

IPX = IR0*IR1; <1>

(1.m+28.n)

(1.m+16.n+12)

IPY = IR0*IR2; <2>

(1.m+28.n)

(1.m+16.n+12)

IPZ = IR0*IR3; <3>

(1.m.n)

(1.m.n)

IR1 = limA1S(**IPX**);

(1.m.n)

(1.m.n)

IR2 = limA2S(**IPY**);

(1.m.n)

(1.m.n)

IR3 = limA3S(**IPZ**);

(1.m+16.n)

(1.m+16.n)

MAC1 = **IPX**;

(1.m+16.n)

(1.m+16.n)

MAC2 = **IPY**(1.m+16.n) (1.m+16.n) **MAC3** = **IPZ**;

(-.8.-)

CD0 <- CD1 <- CD2 <- CODE

(0. 0. 8)

R0 <- R1 <- R2 <- limB1(**IPX**);

(0. 0. 8)

G0 <- G1 <- G2 <- limB2(**IPY**);

(0. 0. 8)

B0 <- B1 <- B2 <- limB3(**IPZ**);

GPF sf**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

GPL sf **Required cycles: 5****Function:** General purpose interpolation**Items specified using arguments:**

Argument	Specified content	Value=0	Value=1
sf	Output format	--	Performs calculations on data shifted 12 bits to the left in the IRn register.

Calculations: (m and n specify the data format of IRp(p=1,2,3) as (1.m.n).)

sf == 0

sf == 1

(1.m+28.n) (1.m+16.n+12) **IPX** = **MAC1** + IR0*IR1; <1>(1.m+28.n) (1.m+16.n+12) **IPY** = **MAC2** + IR0*IR2; <2>(1.m+28.n) (1.m+16.n+12) **IPZ** = **MAC3** + IR0*IR3; <3>(1.m.n) (1.m.n) IR1 = limA1S(**IPX**);(1.m.n) (1.m.n) IR2 = limA2S(**IPY**);(1.m.n) (1.m.n) IR3 = limA3S(**IPZ**);(1.m+16.n) (1.m+16.n) **MAC1** = **IPX**;(1.m+16.n) (1.m+16.n) **MAC2** = **IPY**;(1.m+16.n) (1.m+16.n) **MAC3** = **IPZ**;

(-.8.-) CD0 <- CD1 <- CD2 <- CODE

(0. 0. 8) R0 <- R1 <- R2 <- limB1(**IPX**);(0. 0. 8) G0 <- G1 <- G2 <- limB2(**IPY**);(0. 0. 8) B0 <- B1 <- B2 <- limB3(**IPZ**);

GPL sf**Referenced registers:**

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

Modified registers:

	Data	Control
0	VX0,VY0	R11,R12
1	VZ0	R13,R21
2	VX1,VY1	R22,R23
3	VZ1	R31,R32
4	VX2,VY2	R33
5	VZ2	TRX
6	RGB CODE	TRY
7	OTZ	TRZ
8	IR0	L11,L12
9	IR1	L13,L21
10	IR2	L22,L23
11	IR3	L31,L32
12	SX0,SY0	L33
13	SX1,SY1	RBK
14	SX2,SY2	GBK
15	SX2P,SY2P	BBK
16	SZx(0)	LR1,LR2
17	SZ0(1)	LR3,LG1
18	SZ1(2)	LG2,LG3
19	SZ2(3)	LB1,LB2
20	R0 G0 B0 CD0	LB3
21	R1 G1 B1 CD1	RFC
22	R2 G2 B2 CD2	GFC
23		BFC
24	MAC0	OFX
25	MAC1	OFY
26	MAC2	H
27	MAC3	DQA
28	IRGB	DQB
29	ORGB	ZSF3
30	DATA32	ZSF4
31	LZC	FLAG

