

SOLO1 ASIC Engineering Specification

Part No: 980-03004-F57 WNI-Restricted March 2, 1998



SOLO1 ASIC Engineering Specification

Part No: 980-03004-F57 WNI-Restricted March 2, 1998 Copyright 1998 WebTV Networks, Incorporated. All rights reserved.

This material constitutes confidential and proprietary information of WebTV Networks Incorporated. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written consent of the publisher.

WebTV and WebTV Network are trademarks of WebTV Networks, Inc.

This publication and the information herein is furnished AS IS, is subject to change without notice, and should not be construed as a commitment by WebTV Networks Incorporated. WebTV Networks Incorporated assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied or statutory) with respect to this publication, and expressly disclaims any and all warranties of merchantability, fitness for particular purposes and noninfringement of third party right.

Contents

i illi oduction
1.1 Features1-1
1.1.1 Integrated Video Encoder1-2
1.1.2 Graphics Core1-2
1.1.3 Digital Video Input Port Bus1-2
1.1.4 Memory1-2
1.1.5 Optional Features1-2
1.2 Applications1-3
1.2.1 LC2-BareBones System Features1-3
1.2.2 LC2-Classic System Features1-4
1.2.3 LC2-Deluxe System Features1-5
1.2.4 LC2-Enet System Features1-6
1.2.5 LC2-Sat System Features1-7
1.2.6 LC2-DVD System Features1-8
2 Overview
2.1 SOLO1 System Connections2-3
2.2 New Features2-4
2.2.1 On-chip Video Encoder2-4
2.2.2 Digital Video Input Port2-4
2.2.3 Internal Expansion Bus2-4
2.2.4 Dual Smart Card Interface2-4
2.2.5 Integrated Parallel Port and Serial Ports2-4
2.2.6 Soft Modern Support2-4
2.2.7 IR Receiver and Blaster2-4
2.2.8 Burst mode and Auto-Precharge mode on SDRAM2-4
2.3 Functional Description2-5

2.3.1 CPU Bus Unit	2-5
2.3.2 Memory Unit	2-5
2.3.3 RIO Bus Unit	2-5
2.3.4 Video Unit	2-6
2.3.5 Pixel Output Unit	2-6
2.3.6 Graphics Unit	2-6
2.3.7 Audio Unit	2-6
2.3.8 I/O Device Unit	2-7
2.3.9 Digital Video Input Unit	2-7
2.3.10 Digital Video Encoder Unit	2-7
2.3.11 Modern Unit	
2.3.12 Smart Card and UART Control Unit	2-7
2.3.13 Miscellaneous /Clock Unit	2-8
2.3.14 Other Units	2-8
2.3.14.1 Clock Driver Unit	2-8
2.3.14.2 Side Unit	2-8
2.3.14.3 VO Unit	2-8
2.3.14.4 PLL Unit	2-8
2.3.14.5 DAC Unit	2-8
3 gfxUnit Theory of Operations	
3.1 gfxUnit Overview	3-1
3.2 Object-oriented Graphics	3-1
3.2.1 Compositing	3-2
3.2.2 Cels	3-2
3.2.3 yMap Structure	3-4
3.2.4 Cels Structure	3-4
3.3 TextureMaps	3-5
3.3.1 Direct TextureMap Formats	3-5
3.4 VQ TextureMap and Codebook Formats	3-7
3.4.1 VQ Format Summary	3-9
3.4.1.1 VQ/4:4:4	3-9
3.4.1.2 VQ/42:2 ,	3-11
3.5 Forward Texture Mapping	3-12
4 Video Tuner/Digitizer	
Theory of Operations	
4.1 Overview	4-1
4.2 Design Issues and LC2 Implementation	
end, the account to the comment of the comment of the description of the post 2 and	

4.2.1 Digital Video Interface Unit (divUnit)	4-3
4.2.2 Digital Video Encoder (dveUnit)	
4.2.2.1 Sample Applications	
4.2.2.2 VGA Timing	
4.3 Tuner	4-7
5 Software Description	
5.1 Memory Map	5. 1
5.2 Registers	
5.2.1 Register Summary	
5.2.2 busUnit Software Description	
5.2.2.1 Reset Behavior	
5.2.2.2 Interrupt Handling	
5.2.2.3 Third-level Interrupt Status	
5.2.3 busUnit	
5.2.3.1 BUS_CHIPID Register	
5.2.3.2 BUS_CHPCNTL Register	
5.2.3.3 BUS_INTSTAT Register	
5.2.3.4 BUS_INTEN Registers	
5.2.3.5 BUS_ERRSTAT Registers	
5.2.3.6 BUS_ERREN Registers	
5.2.3.7 BUS_ERRADDR Register	
5.2.3.8 BUS_WDVALUE Register	
5.2.3.9 Fence Registers	
5.2.3.10 BUS_TCOUNT Register	
5.2.3.11 BUS_TCOMPARE Register	
5.2.3.12 BUS_INTSTATRAW Register	
5.2.3.13 BUS_GPINTSTAT Registers	
5.2.3.14 BUS_GPINTEN Registers	
5.2.3.15 BUS_GPINTPOL Register	
5.2.3.16 BUS_AUDINTSTAT Register	
5.2.3.17 BUS_AUDINTEN Registers	
5.2.3.18 BUS_DEVINTSTAT Registers	
5.2.3.19 BUS_DEVINTEN Registers	
5.2.3.20 BUS_VIDINTSTAT Registers	
5.2.3.21 BUS_VIDINTEN Registers	
5.2.3.22 BUS_RIOINTSTAT Registers	
5.2.3.23 BUS_RIOINTPOL Register	
5.2.3.24 BUS_RIOINTEN Registers	

	5.2.3.25 TIMINTSTAT Registers	j-44
	5.2.3.26 TIMINTEN Registers	5-44
	5.2.3.27 BUS_RESETCAUSE Registers	j-45
	5.2.3.28 BUS_J1FENLADDR Register	5-45
	5.2.3.29 BUS_J1FENHADDR Register	5-45
	5.2.3.30 BUS_J2FENLADDR Register	5-46
	5.2.3.31 BUS_J2FENHADDR Register	5-46
	5.2.3.32 BUS_TOPOFRAM Register	5-46
	5.2.3.33 BUS_FENCECNTL Register	5 -47
	5.2.3.34 BOOTMODE Register	5-4 7
	5.2.3.35 USEBOOTMODE Register	5-47
	5.2.3.36 BUS_WDREG_C Register	5-48
5.2.	4 rioUnit Software Description	5-49
	5.2.4.1 ROM Interface	5-49
	5.2.4.2 Reset Behavior	5-51
	5.2.4.3 Asynchronous Device Interface	5-51
	5.2.4.4 Typical System Configuration	5-52
	5.2.4.5 Synchronous Device Interface	5-54
5.2.	5 rioUnit Registers	5-56
	5.2.5.1 RIO_SYSCONFIG Register	5-56
	5.2.5.2 ROM_CNTL0/1 Registers	5-57
	5.2.5.3 DEV_CNTL0/1 Registers	5-57
	5.2.5.4 WTV_CNTL Register	
	5.2.5.5 RIO_CNTL Register	5-59
5.2.	6 audUnit Registers	5-60
	5.2.6.1 AUD_OCSTART Register	5- 6 0
	5.2.6.2 AUD_OCSIZE Register	5-61
	5.2.6.3 AUD_OCCONFIG Register	5-61
	5.2.6.4 AUD_OCCNT Register	5-61
	5.2.6.5 AUD_ONSTART Register	5-62
	5.2.6.6 AUD_ONSIZE Register	5-62
	5.2.6.7 AUD_ONCONFIG Register	5-62
	5.2.6.8 AUD_ODMACNTL Register	5-63
	5.2.6.9 AUD_ICSTART Register	5-63
	5.2.6.10 AUD_ICSIZE Register	5-64
	5.2.6.11 AUD_ICCNT Register	5-64
	5.2.6.12 AUD_INSTART Register	5-64
	5.2.6.13 AUD_INSIZE Register	5-65

5.2.6.14 AUD_IDMACNTL Register	5-65
5.2.6.15 AUD_FCNTL Register	5-65
5.2.6.16 AUD_GPOCNTL Register	5-66
5.2.6.17 AUD_IODMACNTL Register	5-67
5.2.7 vidUnit Registers	5-68
5,2.7.1 vidUnit Programming Overview	5-68
5.2.7.2 Pixel Format	5-70
5.2.7.3 VID_CSTART Register	5-71
5.2.7.4 VID_CSIZE Register	5-71
5.2.7.5 VID_CCNT Register	5-72
5.2.7.6 VID_NSTART Register	5-72
5.2.7.7 VID_NSIZE Register	5-72
5.2.7.8 VID_DMACNTL Register	5-73
5.2.7.9 VID_INTSTAT Register	5-74
5.2.7.10 VID_INTEN Registers	5-75
5.2.7.11 VID_VDATA Register	5-75
5.2.7.12 Typical vidUnit Programming	5-76
5.2.8 devUnit Registers	5-78
5.2.8.1 DEV_LED Register	5-78
5.2.8.2 DEV_IDCNTL Register	5-79
5.2.8.3 DEV_IICCNTL Register	5-79
5.2.8.4 DEV_GPIOIN Register	5-80
5.2.8.5 DEV_GPIOOUT Register	5-80
5.2.8.6 DEV_GPIOEN Register	5-81
5.2.8.7 DEV_RAMDEL Register	5-81
5.2.8.8 DEV_IRIN_SAMPLE Register	5-81
5.2.8.9 DEV_IRIN_REJECT_INT Register	5-82
5.2.8.10 DEV_IRIN_TRANS_DATA Register	5-82
5.2.8.11 DEV_IRIN_STATCNTL Register	5-82
5.2.8.12 IROUT Register Overview	5-83
5.2.8.13 DEV_IROUT_FIFO Register	5-83
5.2.8.14 DEV_IROUT_STATUS Register	5-83
5.2.8.15 DEV_IROUT_PERIOD Register	5-85
5.2.8.16 DEV_IROUT_ON Register	5-85
5.2.8.17 DEV_IROUT_CURRENT_PERIOD Register	5-85
5.2.8.18 DEV_IROUT_CURRENT_ON Register	5-85
5.2.8.19 DEV_IROUT_CURRENT_COUNT Register	5-86
5.2.8.20 Parallel Port Registers	5-86

	5.2.11.2 GRFX_OOTYCOUNT Register	.5-110
	5.2.11.3 GRFX_CELSBASE Register	.5-110
	5.2.11.4 GRFX_YMAPBASE Register	.5-110
	5.2.11.5 GRFX_YMAPBASEMASTER	.5-111
	5.2.11.6 GRFX_INITCOLOR Register	5-112
	5.2.11.7 GRFX_YCOUNTERINIT Register	.5-112
	5.2.11.8 GRFX_PAUSECYCLES Register	.5-113
	5.2.11.9 GRFX_OOTCELSBASE Register	.5-113
	5.2.11.10 GRFX_OOTYMAPBASE Register	.5-113
	5.2.11.11 GRFX_OOTCELSOFFSET Register	.5-114
	5.2.11.12 GRFX_OOTYMAPCOUNT Register	.5-114
	5.2.11.13 GRFX_TERMCYCLECOUNT Register	.5-115
	5.2.11.14 GRFX_HCOUNTERINIT Register	5-115
	5.2.11.15 GRFX_BLANKLINES and GRFX_ACTIVELINES	
	Registers	
	5.2.11.16 GFX_INTEN Register	
	5.2.11.17 GFX_INTSTAT Register	
	5.2.11.18 General Operation of the Write-back Unit	
	5.2.11.19 GFX_WBDSTART Register	
	5.2.11.20 GFX_WBDLSIZE Register	
	5.2.11.21 GFX_WBSTRIDE Register	
	5.2.11.22 GFX_WBDCONFIG Register	
5.2.	12 divUnit Registers	
	5.2.12.1 DIV_SYNCCNTL Register	
	5.2.12.2 DIV_DMACNTL Register	
	5.2.12.3 DIV_NEXTVBITB Register	
	5.2.12.4 DIV_NEXTVBILR Register	
	5.2.12.5 DIV_NEXTVBIADDR Register	
	5.2.12.6 DIV_NEXTTB Register	
	5.2.12.7 DIV_NEXTLR Register	
	5.2.12.8 DIV_NEXTCFG Register	
	5.2.12.9 DIV_NEXTADDR Register	
	5.2.12.10 DIV_CURRVBITB Register	
	5.2.12.11 DIV_CURRVBILR Register	
	5.2.12.12 DIV_CURRVBIADDR Register	.5-126
	5.2.12.13 DIV_CURRTB Register	.5-127
	5.2.12.14 DIV_CURRLR Register	
	5.2.12.15 DIV_CURRCFG Register	
	5.2.12.16 DIV_CURRADDR Register	.5-128

	5.2.12.17 DIV_AUDCNTL Register	5-128
	5.2.12.18 DIV_NEXTAUDADDR Register	5-130
	5.2.12.19 DIV_NEXTAUDLEN Register	5-130
	5.2.12.20 DIV_CURRAUDADDR Register	5-130
	5.2.12.21 DIV_CURRAUDLEN Register	5-130
	5.2.12.22 DIV_SYNCPHASE Register	5-131
	5.2.12.23 DIV_GPOEN Register	5-131
	5.2.12.24 DIV_GPO Register	5-132
	5.2.12.25 DIV_GPI Register	5-132
5.2	.13 dveUnit Registers	5-133
	5.2.13.1 DVE_CNTL Register	5-133
	5.2.13.2 DVE_CNFG Register	5-133
	5.2.13.3 DVE_DBDATA Register	5-135
	5.2.13.4 DVE_DBEN Register	5-135
	5.2.13.5 DVE_DTST Register	5-136
	5.2.13.6 DVE_RDFIELD Register	5-136
	5.2.13.7 DVE_RDPHASE Register	5-136
	5.2.13.8 DVE_FILTCNTL Register	5-136
5.2	.14 potUnit Registers	5-138
	5.2.14.1 POT_VSTART Register	5-139
	5.2.14.2 POT_VSIZE Register	5-139
	5.2.14.3 POT_BLNKCOL Register	5-139
	5.2.14.4 POT_HSTART Register	5-140
	5.2.14.5 POT_HSIZE Register	5-140
	5.2.14.6 POT_CNTL Register	5-140
	5.2.14.7 POT_HINTLINE Register	5-141
	5.2.14.8 POT_INTEN Register	5-142
	5.2.14.9 POT_INTSTAT Register	5-143
	5.2.14.10 POT_CLINE Register	5-144
	5.2.14.11 POT_INTEN_C Register	5-144
	5.2.14.12 POT_INTSTAT_C Register	
	5.2.14.13 Typical potUnit Programming	5-145
5.2	.15 sucUnit Registers	5-147
	5.2.15.1 UART FIFO Registers	
	5.2.15.2 SUCGPU_TFFTRG /SUCSCO_TFFTRG Registers	5-151
	5.2.15.3 Shift Register Registers	
	5.2.15.4 Clock Divider Registers	5-162
	5.2.15.5 Line Control and Status Begisters	5-166

6.2.4.1 State Machine Address Path6-8

6.2.4.2 State Machine Data Path	6-8
6.2.5 RIO Bus Interface	6-9
6.2.6 ISA Device Interface	6-10
6.2.6.1 Holding the Bus for a Slow Device	6-11
6.2.7 WebTV Port Expansion Interface	6-11
6.2.8 Signal Definitions	6-11
6.3 Audio Unit	6-13
6.3.1 Overview	6-13
6.3.2 Internal vs. Codec Data Formats	6-15
6.3.3 Data Out Formats	6-15
6.3.4 Data Interface Formats	6-16
6.3.5 DMA Engine Description	6-17
6.3.6 Signal Definitions	6-17
6.4 Bus Unit	6-20
6.4.1 Overview	6-20
6.4.2 Reset Behavior	6-21
6.4.3 Write Buffer	6-21
6.4.4 Signal Definitions	6-22
6.4.5 Bus Reads	6-23
6.4.6 Bus Writes	6-23
6.5 Digital Video Input Unit	6-25
6.5.1 Overview	6-25
6.5.2 Signal Definitions	6-27
6.5.3 Programming Model	6-27
6.6 Modern Unit	6-29
6.6.1 Overview	6-29
6.6.2 Signal Definitions	6-30
6.7 I/O Device Unit	6-31
6.7.1 Overview	6-31
6.7.2 Signal Definitions	6-32
6.7.3 Parallel Port	6-34
6.7.3.1 Differences Between the SOLO1 and Sasha Handshake	
State Machines	
6.7.3.2 Register Changes	
6.7.3.3 Other Changes	
6.8 Pixel Output Unit	
6.8.1 Overview	
6.8.2 Signal Definitions.	
6.9 Video Unit	6-38

6.9.1 Overview	6-38
6.9.2 Signal Definitions	6-39
6.10 Digital Video Encoder Unit	6-40
6.10.1 Overview	6-40
6.10.2 Signal Definitions	6-41
6.11 Memory Unit	6-43
6.11.1 Overview	6-43
6.11.1.1 Access Technique	6-45
6.11.1.2 Arbitration Algorithm	6-45
6.11.1.3 Memory Refreshing	6-46
6.11.1.4 memUnit Transaction Timing	6-47
6.11.2 Signal Definitions	6-48
6.12 Smart Card and UART Control Unit	6-49
6.12.1 Overview	6-49
6.12.2 Signal Definitions	6-50
6.12.3 Submodule Descriptions	6-50
6.12.3.1 sucUnit Bus Interface Submodule (sucBusIF)	6-50
6.12.3.2 sucUnit UART Submodule (sucUart)	6-50
6.12.3.3 sucUnit Diagnostic Module (sucDiag)	6-53
6.12.4 WebTV Terminal EMV Specification Compliance	6-53
6.12.4.1 EMV Compliance Overview	6-54
6.12.5 Electromechanical Interface (EMV '96 Part I, 1)	6-54
6.12.5.1 Electrical Characteristics of the Terminal (EMV '96 Part I, 1.4)	6-54
6.12.6 Card Session (EMV '96 Part I, 2)	6-56
6.12.6.1 Normal Card Session (EMV '96 Part I, 2.1)	6-56
6.12.6.2 Physical Transportaion of Characters (EMV '96 Part I, 3)	6-59
6.12.6.3 Answer-To-Reset (EMV '96 Part I, 4)	6-60
6.12.6.4 Transmission Protocols (EMV '96 Part I, 5)	6-62
6.13 Graphics Unit	6-63
6.13.1 Overview	6-63
6.13.2 Signal Definitions	6-65
6.13.3 Control Subunit	6-67
6.13.3.1 State Machine	6-67
6.13.3.2 Address Generation	6-68
6.13.3.3 Address Generation Summary	6-70
6.13.4 Mapping Subunit	6-72
6.13.4.1 Mapping Stage 1	6-72
6.13.4.2 Stage 2	6-74

6.13.5 Vector Quantitization Subunit	6-74
6.13.5.1 Vector/Codebook Cache	6-75
6.13.5.2 Codebook	6-76
6.13.6 Blend Subunit	6-76
6.13.6.1 Blending Algorithm	6-76
6.13.6.2 Details of the Multipliers and Adders	6-78
6.13.6.3 Alpha Blend Modes	6-78
6.13.7 RAM Line Buffer Subunit	6-79
6.13.8 Video Output Subunit	6-80
6.13.8.1 Determining Odd or Even Fields	6-81
6.13.8.2 The VCount Counter	6-82
6.13.8.3 TV Display Characteristics	6-83
6.14 Miscellaneous/Clock Unit	6-85
6.14.1 mckTest	6-85
6.14.2 mckResetAll	6-86
6.14.3 mckResetCpu	6-86
6.14.4 Signal Definitions	6-87
6.15 PLL Unit	6-88
6.15.1 Overview	6-88
6.15.2 Signal Definitions	6-89
6.16 Reset Logic	6-90
6.17 Test Modes	6-93
6.17.1 Scan Operation	6-93
6.17.2 Extended Test Modes	6-93
6.18 Electrical Specifications	6-95
6.18.1 Signal Descriptions	6-95
6.18.2 Pin Assignments	6-99
6.19 Mechanical Specifications	6-111
A SOLO1 Rev. 1.3 Errata	
A.1 Overview	A-1
A.2 SDRAM Self-refresh Fix	A-1
A.3 External Master/Slave Device on CPU Interface	A-1
A.4 Enabling the Enhanced CPU Interface	A-1
A.5 Solo and External Device Address Spaces	A-2
A.6 Enhanced Mode CPU Interface Signals	
A.7 Enhanced Mode System Description	A-4
A.8 Transactions with the CPU as Master	A-5
A.9 Transactions with External Device as Master	A-9

A 10. Transactions leaved White Datumine Dead Date to the ODII	* **
A.10 Transactions Issued While Returning Read Data to the CPU	
A.11 Transaction Size Restrictions	A-12
A.12 Transaction Performance Issues	A-13
B SOLO1 Errata Summary	
B.13 Errata Details	B-3
B.14 Video Output	B-3
B.15 Graphics	B-5
B.16 Video Input	B-7
B.17 Memory	B-8
B.18 Test	B-9
B.19 Other	B-9

List of Figures

Figure 1-1	LC2-BareBones Architecture	1-3
Figure 1-2	LC2-Classic Architecture	1-4
Figure 1-3	LC2-Deluxe Architecture	1-5
Figure 1-4	LC2-Enet Architecture	1-6
Figure 1-5	LC2-Sat Architecture	1-7
Figure 1-6	LC2-DVD Architecture	1-8
Figure 2-1	SOLO1 Functional Block Diagram	2-2
Figure 2-2	SOLO1 External Connections and Components	2-3
Figure 3-1	Cel Types	3-2
Figure 3-2	Example Cels	3-3
Figure 3-3	6x7-pixel 4:4:4 Direct TextureMap	3-6
Figure 3-4	12x7-pixel 4:2:2 Direct TextureMap Organization	3-6
Figure 3-5	VQ/4:4:4 Format	3-9
Figure 3-6	VQ/4:2:2 Format	-11
Figure 4-1	Tuner/Digitizer and Supporting Logic	4-3
Figure 4-2	Simultaneous Y/C/Composite Outputs	4-6
Figure 4-3	Analog Output IC Functions	4-6
Figure 4-4	SCART Connector with Composite, Y/C and RGB	4-7
Figure 5-1	WebTV™ Memory Map	5-1
Figure 5-2	ROM Aliasing	-49
Figure 5-3	ROM Aliasing into RAM Space 5	-50
Figure 5-4	Hold the Bus! 5	-58
Figure 5-5	Video Display Overview	-68
Figure 5-6	IR Output Signal	-83
Figure 5-7	Write-back Unit Operation	118
Figure 5-8	Interlaced Frame Scanning	119

Figure 5-9 Video Display 5-138
Figure 5-10 Frame Buffer Mode 5-188
Figure 5-11 GRFX Direct Mode 5-189
Figure 5-12 GRFX Write-back Mode 5-190
Figure 5-13 4:4;4 Pixel Format 5-191
Figure 5-14 4:2:2 Pixel Format
Figure 5-15 Direct vs. VQ Textures
Figure 5-16 Data Structures
Figure 5-17 yMap Structure
Figure 5-18 Cels Structure Numbering
Figure 5-19 Full CelRecord Parameters
Figure 5-20 MiniCelRecord
Figure 5-21 MicroCelRecord
Figure 5-22 Screen Coordinates
Figure 5-23 Texture Coordinates 5-209
Figure 6-1 SOLO1 Functional Block Diagram (only principal data
paths shown)
Figure 6-2 SOLO1 Clock Domains 6-3
Figure 6-3 rioUnit Functional Block Diagram
Figure 6-4 State Diagram of the rioUnit State Machine
Figure 6-5 RIO Bus Timing for a ROM Write
Figure 6-6 RIO Bus Timing for ISA Reads and Writes
Figure 6-7 Hold the Bus!
Figure 6-8 audUnit Functional Block Diagram
Figure 6-9 Internal and Codec Audio Data Formats
Figure 6-10 Right/Left-Aligned Data Formats
Figure 6-11 Right-justified Data Timing 6-16
Figure 6-12 Left-justified Data Timing
Figure 6-13 busUnit Functional Block Diagram
Figure 6-14 divUnit Functional Block Diagram
Figure 6-15 modUnit Functional Block Diagram
Figure 6-16 devUnit Functional Block Diagram
Figure 6-17 potUnit Functional Block Diagram 6-36
Figure 6-18 vidUnit Functional Block Diagram 6-38
Figure 6-19 dveUnit Functional Block Diagram 6-41
Figure 6-20 memUnit Functional Block Diagram 6-44
Figure 6-21 memUnit Transaction Timing
Figure 6-22 sucUnit Functional Block Diagram
Figure 6-23 sucUart Submodule Functional Block Diagram 6-52

Figure 6-24	gfxUnit Functional Block Diagram	6-64
Figure 6-25	State Machine Diagram	6-68
Figure 6-26	Address Generation Block Diagram	6-71
Figure 6-27	Mapping Function Stage 1	6-73
Figure 6-28	Mapping Function Stage 2	6-74
Figure 6-29	Vector Buffer	6-75
Figure 6-30	Two-by-two Pixel Block in VQ8 4:2:2 Codebook	6-76
Figure 6-31	Alpha Blend Block Diagram	6-77
Figure 6-32	Scanline Buffer Organization	6-79
Figure 6-33	One Pixel Composited Every Cycle	6-80
Figure 6-34	Ping-Pong Operation	6-80
Figure 6-35	Pixel Output	6-81
Figure 6-36	Shifting Y, Cb, Cr Values	6-81
Figure 6-37	NTSC Line Numbering	6-82
Figure 6-38	PAL Line Numbering	6-83
Figure 6-39	NTSC/PAL System Timing	6-84
Figure 6-40	pllUnit Functional Block Diagram	6-88
Figure 6-41	Reset Logic	6-92
Figure 6-42	Type 1 Packaging Dimensions	3-111
Figure 6-43	Type 2 Packaging Dimensions	5-113
Figure A-1	System Address Maps	A-3
Figure A-2	System Block Diagram	A-5
Figure A-3	CPU Write to SOLO1	A-6
Figure A-4	CPU Write to an External Device	A-7
Figure A-5	CPU Read from SOLO1	A-8
Figure A-6	CPU Read from an External Device	A-9
Figure A-7	External Device Write to SOLO1	A-10
Figure A-8	External Device Single-word Read from SOLO1	A-11
•	External Device Write to SOLO1 Underneath	
Read Response	onse	A-12

List of Tables

Table 3-1	gfxunit Formats	3-9
Table 4-1	Tuner/Digitizer Operating Modes	4-2
Table 4-2	DVE Output Mode Signal Levels	4-4
Table 5-1	SOLO1 Registers - Arranged by Address	5-2
Table 5-2	Interrupt Status and Enable Registers	5-20
Table 5-3	Second-level Interrupt Registers	5-21
Table 5-4	Third-level Interrupt Registers	5-23
Table 5-5	Typical System Interrupt Configuration	5-23
Table 5-6	BUS_CHIPID Bit Definitions (0x0400_0000)	5-29
Table 5-7	BUS_CHPCNTL Bit Definitions (0x0400_0004)	5-29
	BUS_INTSTAT Bit Definitions (0x0400_0008, 0x0400_0050, 108)	5-30
Table 5-9	BUS_INTEN Bit Definitions (0x0400_000C, 10C)	
0x0400_0	D BUS_ERRSTAT Bit Definitions (0x0400_0010, 110, 054)	5-31
Table 5-1	1 BUS_ERREN Bit Definitions (0x0400_0014, 114)	
	2 BUS_ERRADDR Bit Definitions (0x0400_0018)	
	BUS_WDVALUE Bit Definitions (0x0400_0030)	
Table 5-1	4 BUS_LOWRDADDR Bit Definitions (0x0400_0034)	5-34
	5 BUS_LOWRDMASK Bit Definitions (0x0400_0038)	
Table 5-1	6 BUS_LOWRDADDR Bit Definitions (0x0400_003C)	5-35
Table 5-1		
Table 5-1	B BUS_TCOUNT Bit Definitions (0x0400_0048)	
	9 BUS_TCOMPARE Bit Definitions (0x0400_004C)	
	0 BUS_INTSTATRAW Bit Definitions (0x0400_0050)	

	BUS_ GPINTSTAT Bit Definitions (0x0400_0058, 0x0400_0060, 8)	5-37
-	BUS_ GPINTPEN Bit Definitions (0x0400_005C,	
	C)	5-38
Table 5-23	BUS_ GPINTPOL Bit Definitions (0x0400_0064)	5-39
	BUS_AUDINTSTAT Bit Definitions (0x0400_0068, 0x0400_006C, 8)	5-40
Table 5-25	BUS_AUDINTEN Bit Definitions (0x0400_0070,	
0x0400_0170	0)	5-40
	BUS_ DEVINTSTAT Bit Definitions (0x0400_0074, 0x0400_078, 4)	5-41
Table 5-27	BUS_ DEVINTEN Bit Definitions (0x0400_007C,	
0x0400_0170	C)	5-41
	BUS_ VIDINTSTAT Bit Definitions (0x0400_0080, 0x0400_084, 0)	5-42
	BUS_VIDINTEN Bit Definitions (0x0400_0088,	
	8)	5-42
Table 5-30	BUS_ RIOINTSTAT Bit Definitions (0x0400_008C, 0x0400_0090,	
	C)	
	BUS_ RIOINTPOL Bit Definitions (0x0400_0094)	5-43
	BUS_RIOINTPEN Bit Definitions (0x0400_0098,	
	8)	5-44
	BUS_TIMINTSTAT Bit Definitions (0x0400_009C,	
	, 0x0400_019C)	5-44
	BUS_TIMINTEN Bit Definitions (0x0400_00A4, 4)	5-44
	BUS_ RESETCAUSE Bit Definitions (0x0400_00a8, C)	5-45
Table 5-36	BUS_J1FENLADDR Bit Definitions (0x0400_00B0)	5-45
Table 5-37	BUS_J1FENHADDR Bit Definitions (0x0400_00B4)	5-45
Table 5-38	BUS_J2FENLADDR Bit Definitions (0x0400_00B8)	5-46
Table 5-39	BUS_J2FENHADDR Bit Definitions (0x0400_00BC)	5-46
Table 5-40	BUS_TOPOFRAM Bit Definitions (0x0400_00C0)	5-46
Table 5-41	BUS_FENCECNTL Bit Definitions (0x0400_00C4)	5-47
Table 5-42	BUS_ BOOTMODE Bit Definitions (0x0400_00C8)	5-47
Table 5-43	BUS_ USEBOOTMODE Bit Definitions (0x0400_00CC)	5-47
Table 5-44	BUS_WDREG_C Bit Definitions (0x0400_0118)	5-48
Table 5-45	RIO_SYSCONFIG Bit Definitions (0x0400_1000)	5-56
Table 5-46	ROM_CNTL0/1 Bit Definitions (0x0400_1004,	
0x0400 100	8)	5-57

Table 5-47	DEV_CNTL0-1 Bit Definitions (0x0400_100C,		
0x0400_1010)5-57			
Table 5-48	WTV_CNTL Bit Definitions (0x0400_101C)5-58		
Table 5-49	RIO_CNTL Bit Definitions (0x0400_1020)5-59		
Table 5-50	AUD_OCSTART Bit Definitions (0x0400_20005-60		
Table 5-51	AUD_OCSIZE Bit Definitions (0x0400_2004)5-61		
Table 5-52	AUD_OCCONFIG Bit Definitions (0x0400_2008)5-61		
Table 5-53	AUD_OCCNT Bit Definitions (0x0400_200C)5-61		
Table 5-54	AUD_ONSTART Bit Definitions (0x0400_2010) 5-62		
Table 5-55	AUD_ONSIZE Bit Definitions (0x0400_2014) 5-62		
Table 5-56	AUD_ONCONFIG Bit Definitions (0x0400_2018)5-62		
Table 5-57	AUD_ODMACNTL Bit Definitions (0x0400_201C)5-63		
Table 5-58	AUD_ICSTART Bit Definitions (0x0400_2020)5-63		
Table 5-59	AUD_ICSIZE Bit Definitions (0x0400_2024)5-64		
Table 5-60	AUD_ICCNT Bit Definitions (0x0400_202C)5-64		
Table 5-61	AUD_INSTART Bit Definitions (0x0400_2030)5-64		
Table 5-62	AUD_INSIZE Bit Definitions (0x0400_2034)5-65		
Table 5-63	AUD_IDMACNTL Bit Definitions (0x0400_203C) 5-65		
Table 5-64	AUD_FCNTL Bit Definitions (0x0400_2040)5-65		
Table 5-65	AUD_GPOCNTL Bit Definitions (0x0400_2044)5-66		
Table 5-66	AUD_IODMACNTL Bit Definitions (0x0400_205C)5-67		
Table 5-67	Pixel Format in Memory5-71		
Table 5-68	VID_CSTART Bit Definitions (0x0400_3000)5-71		
Table 5-69	VID_CSIZE Bit Definitions (0x0400_3004)5-71		
Table 5-70	VID_CCNT Bit Definitions (0x0400_3008)5-72		
Table 5-71	VID_NSTART Bit Definitions (0x0400_300C) 5-72		
Table 5-72	VID_NSIZE Bit Definitions (0x0400_3010)5-73		
Table 5-73	VID_DMACNTL Bit Definitions (0x0400_3014)5-73		
Table 5-74	VID_INTSTAT Bit Definitions (0x0400_3038,		
0x0400_31	38)5-74		
Table 5-75	VID_ INTEN Bit Definitions (0x0400_303C,		
0x0400_31	3C)5-75		
Table 5-76	VID_ INTEN Bit Definitions (0x0400_3040)5-75		
Table 5-77	DEV_IROLD Bit Definition (0x0400_4000)5-78		
Table 5-78	DEV_LED Bit Definition (0x0400_4004)5-78		
Table 5-79	DEV_IDCNTL Bit Definition (0x0400_4008)5-79		
Table 5-80	DEV_IICCNTL Bit Definition (0x0400_400C)5-79		
Table 5-81	DEV_GPIOIN Bit Definition (0x0400_4010)5-80		

Table 5-82 DEV_GPIOOUT Bit Definition (0x0400_4014,	
0x0400_4114)	5-80
Table 5-83 DEV_GPIOEN Bit Definition (0x0400_4018, 0x0400_4118)	5-81
Table 5-84 DEV_RAMDEL Bit Definition (0x0400_4800)	5-81
Table 5-85 DEV_IRIN_SAMPLE Bit Definition (0x0400_4020)	5-81
Table 5-86 DEV_IRIN_REJECT_INT Bit Definition (0x0400_4024)	5-82
Table 5-87 DEV_IRIN_TRANS_DATA Bit Definition (0x0400_4028)	5-82
Table 5-88 DEV_IRIN_STATCNTL Bit Definition (0x0400_402C)	5-82
Table 5-89 DEV_IROUT_FIFO Bit Definition (0x0400_4040)	5-83
Table 5-90 DEV_IROUT_STATUS Bit Definitions (0x0400_4044)	5-83
Table 5-91 DEV_IROUT_PERIOD Bit Definition (0x0400_4048)	5-85
Table 5-92 DEV_IROUT_ON Bit Definition (0x0400_404C)	5-85
Table 5-93 DEV_IROUT_CURRENT_PERIOD Bit Definition (0x0400_4050)	5-85
Table 5-94 DEV_IROUT_CURRENT_ON Bit Definition	
(0x0400_4054)	5-85
Table 5-95 DEV_IROUT_CURRENT_COUNT Bit Definition	
(0x0400_4058)	5-86
Table 5-96 DEV_PPORT_DATA Register (0x0400_4200)	5-86
Table 5-97 DEV_PPORT_CTRL Register (0x0400_4204)	5-87
Table 5-98 DEV_PPORT_STAT Register (0x0400_4208)	5-88
Table 5-99 DEV_PPORT_CNFG Register (0x0400_420C)	5-89
Table 5-100 DEV_PPORT_FIFOCTRL Register (0x0400_4210)	5-89
Table 5-101 DEV_PPORT_FIFOSTAT Register (0x0400_4214)	5-90
Table 5-102 DEV_PPORT_TIMEOUT Register (0x0400_4218)	5-90
Table 5-103 DEV_PPORT_STAT2 Register (0x0400_421C)	5-90
Table 5-104 DEV_PPORT_IEN (Interrupt Enable) Register	
(0x0400_4220)	5-91
Table 5-105 DEV_PPORT_IST (Interrupt Status) Register (0x0400_4224)	5-91
Table 5-106 DEV_PPORT_CLRINT Register (0x0400_4228)	5-92
Table 5-107 DEV_PPORT_ENABLE Register (0x0400_422C)	5-93
Table 5-108 Debug Bus Data and the Corresponding	
External Pins	5-93
Table 5-109 DEV_DIAG Bit Definitions (0x0400_4804)	5-94
Table 5-110 DEV_DEVDIAG Bit Definitions (0x0400_4808)	5-94
Table 5-111 MOD_OCSTART Bit Definitions (0x0400_B000	5-95
Table 5-112 MOD_OCSIZE Bit Definitions (0x0400_B004)	5-95

Table 5-113	MOD_OCCONFIG Bit Definitions (0x0400_B008)	. 5-95
Table 5-114	MOD_OCCNT Bit Definitions (0x0400_B00C)	5-96
Table 5-115	MOD_ONSTART Bit Definitions (0x0400_B010)	5-96
Table 5-116	MOD_ONSIZE Bit Definitions (0x0400_B014)	5-9 6
Table 5-117	MOD_ONCONFIG Bit Definitions (0x0400_B018)	5-97
Table 5-118	MOD_ODMACNTL Bit Definitions (0x0400_B01C)	5-97
Table 5-119	MOD_ICSTART Bit Definitions (0x0400_B020	5-98
Table 5-120	MOD_ICSIZE Bit Definitions (0x0400_B024)	5-98
Table 5-121	MOD_ICCNT Bit Definitions (0x0400_B02C)	5-98
Table 5-122	MOD_INSTART Bit Definitions (0x0400_B030)	5-99
Table 5-123	MOD_INSIZE Bit Definitions (0x0400_B034)	5-99
Table 5-124	MOD_IDMACNTL Bit Definitions (0x0400_B03C)	5-99
Table 5-125	MOD_FCNTL Bit Definitions (0x0400_B040)	5-100
Table 5-126	MOD_GPOCNTL Bit Definitions (0x0400_B044)	5-101
Table 5-127	MOD_IODMACNTL Bit Definitions (0x0400_B05C)	5-101
Table 5-128	System Frequency vs. SDRAM Speed	5-102
Table 5-129	MEM_TIMING Bit Definition (0x0400_5000)	5-102
Table 5-130	MEM_BURP Bit Definition (0x0400_5008)	5-103
Table 5-131	MEM_REFCTRL Bit Definition (0x0400_500C)	5-105
Table 5-132	MEM_CMD Bit Definition (0x0400_5010)	5-107
Table 5-133	MEMCMD Encoding	5-107
Table 5-134	GFX_CONTROL Bit Definition (0x0400_6004)	5-109
Table 5-135	GRFX_OOTYCount Bit Definition (0x0400_6010)	5-110
Table 5-136	GRFX_CelsBase Bit Definition (0x0400_6014)	5-110
Table 5-137	GRFX_YMapBase Bit Definition (0x0400_6018)	5-110
Table 5-138	GRFX_CelsBaseMaster Bit Definition	
(0x0400_601	ic)	5-111
Table 5-139	GRFX_YMapBaseMaster Bit Definition	
(0x0400_602	20)	5-111
Table 5-140	GRFX_InitColor Bit Definition (0x0400_6024)	5-112
Table 5-141	GRFX_YCounterInit Bit Definition (0x0400_6028)	5-112
Table 5-142	'GRFX_PauseCycles Bit Definition (0x0400_602c)	5-113
Table 5-143	GRFX_OOTCelsBase Bit Definition (0x0400_6030)	5-113
Table 5-144	GRFX_OOTYMapBase Bit Definition (0x0400_6034)	5-113
Table 5-145	GRFX_OOTCelsOffset Bit Definition (0x0400_6038)	5-114
Table 5-146	GRFX_OOTYMapCount Bit Definition	
7	3c)	5-114
	GRFX_TermCycleCount Bit Definition	
(0x0400_604	40)	. 5-115

Table 5-148	GRFX_HCounterInit Bit Definition (0x0400_6044)	5-115
Table 5-149	GRFX_BlankLines Bit Definition (0x0400_6048)	5-116
Table 5-150	GRFX_ActiveLines Bit Definition (0x0400_604c)	5-116
Table 5-151	GFX_INTEN Bit Definitions (0x0400_6060,	
0x0400_6064	4)	5-116
Table 5-152	GFX_INSTAT Bit Definitions (0x0400_6068,	
0x0400_6060	C)	5-117
Table 5-153	WBDSTART Bit Definitions (0x0400_6080)	5-119
Table 5-154	WBDLSize[9:0] Bit Definitions (0x0400_6084)	5-120
Table 5-155	WBSTRIDE Bit Definitions (0x0400_608C)	5-120
Table 5-156	WBDCONFIG Bit Definitions (0x0400_6090)	5-120
Table 5-157	DIV_SYNCCNTL Bit Definitions (0x0400_8000)	5-122
Table 5-158	DIV_DMACNTL Bit Definitions (0x0400_8004)	5-122
Table 5-159	DIV_NEXTVBITB Bit Definitions (0x0400_8008)	5-123
Table 5-160	DIV_NEXTVBILR Bit Definitions (0x0400_800C)	5-123
Table 5-161	DIV_NEXTVBIADDR Bit Definitions (0x0400_8010)	5-124
Table 5-162	DIV_NEXTTB Bit Definition (0x0400_8014)	5-124
Table 5-163	DIV_NEXTLR Bit Definitions (0x0400_8018)	5-124
Table 5-164	DIV_NEXTCFG Bit Definitions (0x0400_801C)	5-125
Table 5-165	DIV_NEXTADDR Bit Definitions (0x0400_8020)	5-125
Table 5-166	DIV_CURRVBITB Bit Definitions (0x0400_8024)	5-126
Table 5-167	DIV_CURRVBILR Bit Definitions (0x0400_8028)	5-126
Table 5-168	DIV_CURRVBIADDR Bit Definitions (0x0400_802C)	5-126
Table 5-169	DIV_CURRTB Bit Definitions (0x0400_8030)	5-127
Table 5-170	DIV_CURRLR Bit Definitions (0x0400_8034)	5-127
Table 5-171	DIV_CURRCFG Bit Definitions (0x0400_8038)	5-127
Table 5-172	DIV_CURRADDR Bit Definitions (0x0400_803C)	5-128
Table 5-173	DIV_AUDCNTL Bit Definitions (0x0400_8040)	5-128
Table 5-174	DIV_NEXTAUDADDR Bit Definitions (0x0400_8044)	5-130
Table 5-175	DIV_NEXTAUDLEN Bit Definitions (0x0400_8048)	5-130
Table 5-176	DIV_CURRAUDADDR Bit Definitions (0x0400_804C)	5-130
Table 5-177	DIV_CURRAUDLEN Bit Definitions (0x0400_8050)	5-130
Table 5-178	DIV_SYNCPHASE Bit Definitions (0x0400_8060)	5-131
Table 5-179	DIV_GPOEN Bit Definitions (0x0400_8064)	5-131
Table 5-180	DIV_GPO Bit Definitions (0x0400_8068)	5-132
Table 5-181	DIV_GPI Bit Definitions (0x0400_806c)	5-132
Table 5-182	DVE_CNTL Bit Definitions (0x0400_7000)	5-133
Table 5-199	DVE CNEC Bit Definitions (0v0400, 7004)	E 122

Table 5-184	DVE_DBDATA Bit Definitions (0x0400_7008)	5-135
Table 5-185	DVE_DBEN Bit Definitions (0x0400_700C)	. 5-135
Table 5-186	DVE_DTST Bit Definitions (0x0400_7010)	. 5-136
Table 5-187	DVE_RDFIELD Bit Definitions (0x0400_7014)	. 5-136
Table 5-188	DVE_RDPHASE Bit Definitions (0x0400_7018)	5-136
Table 5-189	DVE_FILTCNTL Bit Definitions (0x0400_701C)	5-136
Table 5-190	POT_VSTART Bit Definitions (0x0400_9080)	5-139
Table 5-191	POT_VSIZE Bit Definitions (0x0400_9084)	. 5-139
Table 5-192	POT_BLNKCOL Bit Definitions (0x0400_9088)	. 5-139
Table 5-193	POT_HSTART Bit Definitions (0x0400_908C)	. 5-140
Table 5-194	POT_HSIZE Bit Definitions (0x0400_9090)	. 5-140
Table 5-195	POT_ CNTL Bit Definitions (0x0400_9094)	. 5-140
Table 5-196	POT_ HINTLINE Bit Definitions (0x0400_9098)	. 5-141
Table 5-197	POT_ INTEN Bit Definitions (0x0400_909C)	. 5-142
Table 5-198	POT_ INTSTAT Bit Definitions (0x0400_90A0)	. 5-143
Table 5-199	POT_ CLINE Bit Definitions (0x0400_90AC)	. 5-144
Table 5-200	POT_ INTEN_C Bit Definitions (0x0400_90A4)	. 5-144
Table 5-201	POT_ INTSTAT_C Bit Definitions (0x0400_90a8)	. 5-144
	SUCGPU_TFFHR/SUCSC0_TFFHR Bit Definitions	
	00, 0x0400_A800)	. 5-148
	SUCGPU_ RFFHR/SUCSC0_RFFHR Bit Definitions 40,0x0400_A840)	. 5-149
Table 5-204	SUCGPU_TFFHRSRW/RFFHRSRW, SUCSCO_TFFHRSRW/RFFH	
Definitions (0	0x0400_A004, 0x0400_A804 and 0x0400_A044,	
	4)	. 5-150
	SUCGPU_TFFTRG/SUCSC0_TFFTRG Bit Definitions 08,0x0400_A808)	E 1E1
	SUCGPU_ RFFTRG/SUCSC0_ RFFTRG Bit Definitions	. 5-151
	48,0x0400_A848)	. 5-152
Table 5-207	SUCGPU_TFFCNT/RFFCNT and SUCSCO_TFFCNT/RFFCNT Bit I	Definitions
	0C,0x0400_A04C and 0x0400_A80C,0x0400_A84C)	
	SUCGPU_TFFMAX/RFFMAX and SUCSCO_TFFMAX/RFFMAX Bit	
	10, 0x0400_A050 and 0x0400_A810, 0x0400_A850)	
	SUCGPU_TFFCR/RFFCR and SUCSC0_TFFCR/RFFCR Bit Definit 14, 0x0400_A054 and 0x0400_A814, 0x0400_A854)	
	SUCGPU_TFFSR/RFFSR and SUCSCO_TFFSR/RFFSR Bit Definit	
	18, 0x0400_A058 and 0x0400_A818, 0x0400_A858)	
	SUCGPU_TFFGCR/RFFGCR and SUCSC0_TFFGCR/RFFGCR Bit	
	1C, 0x0400_A05C and 0x0400_A81C, 0x0400_A85C)	
	SUCGPU_TSRCR/SUCSC0_TSRCR Bit Definitions (0x0400_A080,	
0x0400_A88	0)	. 5-157

Table 5-213 SUCGPU_ RSRCR/SUCSC0_ RSRCR Bit Definitions (0x0400_A0C0, 0x0400_A8C0)5-158
Table 5-214 SUCGPU_TSRSTATE/SUCSCO_TSRSTATE Bit Definitions (0x0400_A084, 0x0400_A884)5-159
Table 5-215 SUCGPU_ RSRSTATE/SUCSCO_ RSRSTATE Bit Definitions (0x0400_A0C4, 0x0400_A8C4)5-159
Table 5-216 SUCGPU_TSRBCCNT/SUCSCO_TSRBCCNT Bit Definitions (0x0400_A088, 0x0400_A888)5-160 Table 5-217 SUCGPU_RSRBCCNT/SUCSCO_RSRBCCNT Bit Definitions (0x0400_A0C8,
Table 5-217 SUCGPU_ RSRBCCNT/SUCSCO_ RSRBCCNT Bit Definitions (0x0400_A0C8, 0x0400_A8C8)
0x0400_A88C)5-161
Table 5-219 SUCGPU_TSRBITCNT/SUCSCO_TSRBITCNT Bit Definitions (0x0400_A0CC, 0x0400_A8CC)5-161
Table 5-220 SUCGPU_ MCD0/SUCSCO_ MCD0 Bit Definitions (0x0400_A100, 0x0400_A900)5-162
Table 5-221 SUCGPU_ SCD0/SCD1, SUCSC0_ SCD0/SCD1 Bit Definitions (0x0400_A104, 0x0400_A108, 0x0400_A904, 0x0400_A908)5-163
Table 5-222 SUCGPU_ CCR/SUCSCO_CCR Bit Definitions (0x0400_A10C, 0x0400_A90C)5-165
Table 5-223 SUCGPU_ LCR/SUCSCO_ LCR Bit Definitions (0x0400_A180, 0x0400_A980)5-166
Table 5-224 SUCGPU_LSCR/SUCSC0_LSCR Bit Definitions (0x0400_A184, 0x0400_A984)5-168
Table 5-225 SUCGPU_ LSTPBITS/SUCSCO_LSTPBITS Bit Definitions (0x0400_A188, 0x0400_A988)5-169
Table 5-226 SUCGPU_ LSR/SUCSCO_ LSR Bit Definitions (0x0400_A18C, 0x0400_A98C)5-169
Table 5-227 SUCGPU_ISR/ISR_R, SUCSCO_ISR/ISR_R Bit Definitions (0x0400_A200, 0x0400_AA00 and 0x0400_A204, 0x0400_AA04)5-172
Table 5-228 SUCGPU_IER/_S/_C, SUCSCO_IER/_S/_C Bit Definitions (0x0400_A210, 0x0400_A208, 0x0400_A20C and 0x0400_AA10, 0x0400_AA08, 0x0400_AA0C)5-174
Table 5-229 SUCGPU_ SDSMCR/SUCSC0_SDSMCR Bit Definitions (0x0400_A280, 0x0400_AA80)5-175
Table 5-230 SUCGPU_SDSMSTATE/SUCSCO_SDSMSTATE Bit Definitions (0x0400_A284, 0x0400_AA84)5-176
Table 5-231 SUCGPU_ IOOD/SUCSCO_IOOD Bit Definitions (0x0400_A288, 0x0400_AA88)5-177
Table 5-232 SUCGPU_ SPIOCR/SUCSC0_SPIOCR Bit Definitions (0x0400_A28C, 0x0400_AA8C)5-178
Table 5-233 SUCGPU_ SPIOEN/_C/_S, SUCSCO_ SPIOEN/_C/_S Bit Definitions (0x0400_A298, 0x0400_A294, 0x0400_A290, 0x0400_AA98,
0x0400 AA94, 0x0400 AA90)5-179

Table 5-234 SUCGPU_ GPIODIR/_C/_S, SUCSCO_ GPIODIR/_C/_S Bit Definitions (0x0400_A2A8, 0x0400_A2A4, 0x0400_A2A0, 0x0400_AAA8, 0x0400_AAA4, 0x0400_AAA0)
Table 5-235 SUCGPU_GPIOVAL/_C/_S, SUCSCO_GPIOVAL/_C/_S Bit Definitions (0x0400_A2B8, 0x0400_A2B4, 0x0400_A2B0, 0x0400_AAB8, 0x0400_AAB4,
0x0400_AAB0)
Table 5-236 SUCGPU_GPIORIER/_C/_S, SUCSCO_ GPIORIER/_C/_S, SUCGPU_ GPIOF ER/_C/_S, SUCSCO_ GPIOFIER/_C/_S Bit Definitions (0x0400_A2C8, 0x0400_A2C4, 0x0400_A2C0, 0x0400_AAC8, 0x0400_AAC4, 0x0400_AAC0, 0x0400_A2D8, 0x0400_A2D4 0x0400_A2D0, 0x0400_AAD8, 0x0400_AAD4, 0x0400_AAD0)
Table 5-237 SUCGPU_ GPIOISR/_C/_S, SUCGPU_ GPIORISR, SUCGPU_ GPIOFISR, SUCSCO_ GPIOISR/_C/_S, SUCSCO_ GPIORISR, SUCSCO_ GPIOFISR Bit Definitions (0x0400_A2E8, 0x0400_A2E4, 0x0400_A2E0, 0x0400_A2EC, 0x0400_A2F0, 0x0400_AAE8 0x0400_AAE4, 0x0400_AAE0), 0x0400_AAF0
Table 5-238 SUCGPU_DIAGMODE/SUCSCO_DIAGMODE Bit Definitions (0x0400_A300, 0x0400_AB00)
Table 5-239 Texture Format Summary5-193
Table 5-240 yMap Structure
Table 5-241 CelDisable Bit Definition
Table 5-242 CelRecord Parameters
Table 5-243 Summary of LoadData microCelRecords5-206
Table 6-1 rioUnit Signals6-11
Table 6-2 audUnit Signals6-17
Table 6-3 busUnit Interface Signals6-22
Table 6-4 divUnit Interface Signals6-27
Table 6-5 divUnit Control Registers6-27
Table 6-6 modUnit Interface Signals6-32
Table 6-7 Port Modes6-34
Table 6-8 potUnit Interface Signals6-37
Table 6-9 vidUnit Interface Signals
Table 6-10 dveUnit Interface Signals
Table 6-11 memUnit's Supported Internal Ports
Table 6-12 memUnit Interface Signals
Table 6-13 sucUnit Signals6-50
Table 6-14 sucUart Interface Modes6-51
Table 6-15 diagbus Output Pins6-53
Table 6-16 gfxUnit Interface Signals
Table 6-17 Address Generation Formula
Table 6-18 Alpha Blend Modes6-78
Table 6-19 mckTest Test Modes and Enable Signals6-85
Table 6-20 CPU Reset States6-86

Table 6-21	mckUnit Signals	6-87
Table 6-22	pilUnit Signals	6-89
Table 6-23	SOLO1 Signal Electrical Parameters	6-95
Table 6-24	SOLO1 Pin Assignments	6-99
Table 6-25	SOLO1 Signal Descriptions	6-106
Table 6-26	Type 1 Packaging Dimensions	6-112
Table B-1	Summary of SOLO1 Problems	B-1

Introduction

SOLO1 is the principal system ASIC in all of WebTV Networks' products, starting in the second half of 1997.

The primary design goals of SOLO1 are:

- Support system cost reduction by integrating system components and supporting multiple sources of CPUs
- · Provide on-board printing capability
- Deliver enhanced video effects via the graphics engine
- Allow digital video input to be integrated into the system
- Provide more flexible expansion and system integration options

SOLO1 contains approximately 300,000 random logic gates and 40Kbits of static RAM. SOLO1 is implemented in a 0.30um triple-metal layer process and is in a 420-pin tape ball grid array (TBGA) package.

1.1 Features

SOLO1 includes the following features:

- Integrated CPU, ROM, Flash, SDRAM, and I/O interfaces
- Audio DMA channels for both input and output
- Video output DMA channel for frame buffer display
- High-speed SDRAM memory interface (330MB/sec peak)
- Miscellaneous I/O device interfaces
- High-performance graphics and video effects engine
- Integrated video encoder
- Digital Video Input port
- Hardware support for soft modem
- Integrated parallel and serial ports
- Dual SmartCard interfaces
- I²C interface

- Supports page-mode ROMs
- RIO bus provides a glueless port to many optional I/O interfaces
- 2W power dissipation

1.1.1 Integrated Video Encoder

- Supports YUV 4:2:2 and 4:4:4 formats
- NTSC and PAL
- RGB outputs also available in encoder
- Supports an external video encoder as "back up"

1.1.2 Graphics Core

- 2-D acceleration
- Multiple line buffers
- Flicker elimination in hardware
- Scanline compositing and image decompression on the fly
- "Write-back" mode for extremely complex screens (video games)
- Pixel masking
- Global Alpha value available
- 8-bit Alpha value on a per-pixel basis
- Cel scaling in hardware
- · Cel rotation in hardware

1.1.3 Digital Video Input Port Bus

- Input for digital video
- DMA engine for transfer to memory
- Can manipulate video via the graphics engine
- 720->640 and 720->768 resampling in hardware
- Separate audio input combined with Digital Video Input port for tuner or MPEG audio

1.1.4 Memory

- Support for 0-8MB of Mask ROM (MROM) including page-mode parts
- Support for 0-8MB of Flash memory (NOR Flash)
- Support for 4MB-8MB of SDRAM using 16M-bit technology
- Support for 16MB-64MB of SDRAM using 64M-bit technology

1.1.5 Optional Features

- IEEE 1394 (Firewire) interface via RIO Bus
- USB interface (directly interfaces to RIO Bus)
- Ethernet interface (directly interfaces to RIO Bus)
- ISDN (directly interfaces to RIO Bus)
- Serial E²ROM NVRAM (directly interfaces to SOLO1)
- IDE interface (directly interfaces to RIO Bus)

1.2 Applications

This section describes the various system configurations that use the SOLO1 ASIC.

1.2.1 LC2-BareBones System Features

The main goal of the LC2-BareBones system is lower cost. The following figure illustrates the LC2-BareBones system's architecture.

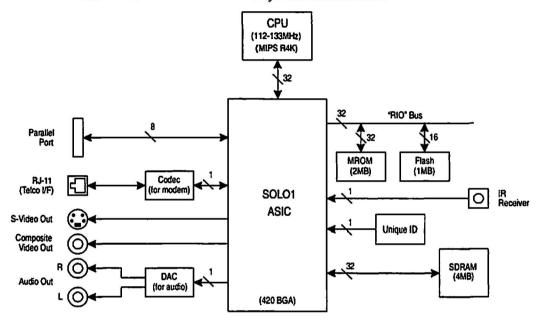


Figure 1-1 LC2-BareBones Architecture

1.2.2 LC2-Classic System Features

The main goal of the LC2-Classic system is to provide the same features as the FCS/LC1 systems for lower cost. Figure 1-2 illustrates the LC2-Classic system's architecture.

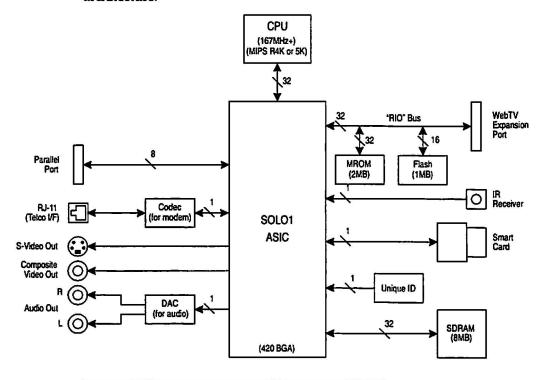


Figure 1-2 LC2-Classic Architecture

The LC2-Classic system is the same as WebTV's 1996 reference platform (FCS/LC1) in terms of features. The only exceptions are that the hard-wired keyboard interface has been removed, the parallel port has been integrated, and there is more RAM.

1.2.3 LC2-Deluxe System Features

The LC2-Deluxe system combines features which enable the further integration of the Web-browsing and TV-watching experience. This integration is accomplished through the addition of an RF tuner and digitizer. Figure 1-3 illustrates the LC2-Deluxe system's architecture.

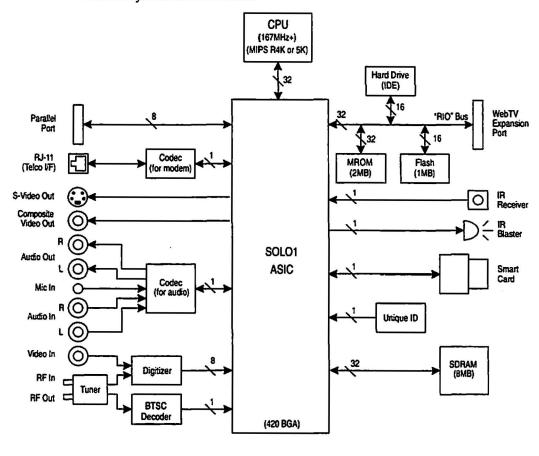


Figure 1-3 LC2-Deluxe Architecture

1.2.4 LC2-Enet System Features

The LC2-Enet system is identical to the "Classic" system, except that the modem subsystem has been replaced with either a 10Base-T or 100Base-T Ethernet subsystem. This increases the network bandwidth to 10M-bits or 100M-bits per second, respectively, as well as reducing the overall system cost.

Note: The RIO bus can also support ISDN chip sets directly.

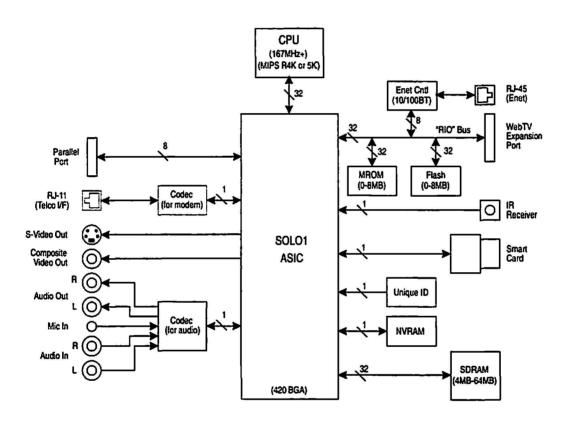


Figure 1-4 LC2-Enet Architecture

1.2.5 LC2-Sat System Features

The LC2-Sat system integrates WebTV technology into a digital satellite system. The divUnit bus can accept the output from the MPEG2 decoder, and the hard drive allows the elimination of the Flash memory (see Figure 1-5).

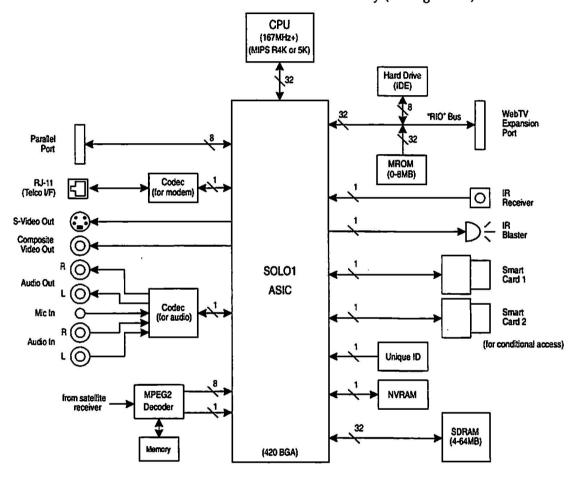


Figure 1-5 LC2-Sat Architecture

1.2.6 LC2-DVD System Features

The LC2-DVD system integrates WebTV technology into a DVD player. Through the divUnit port, the decoded MPEG2 data can be manipulated or passed through directly (see Figure 1-6).

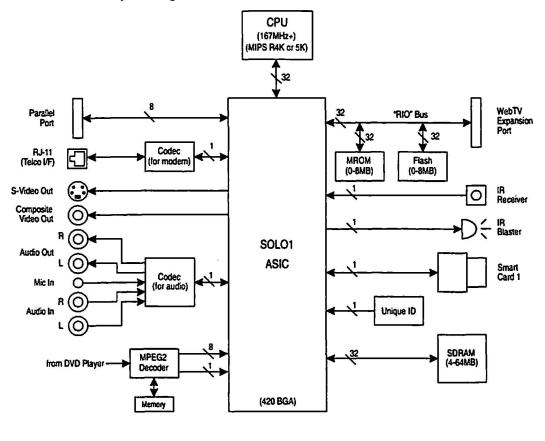


Figure 1-6 LC2-DVD Architecture

Overview

The SOLO1 chip consists of a complete set of I/O interfaces, a graphics engine, video display logic, clock circuits and other miscellaneous logic. The I/O interfaces connect SOLO1 to each of the other components that make up the WebTV system, and allow those components to communicate with each other. The graphics engine, in conjunction with the video display logic, ensures that complex screens can be displayed without exceeding the timing parameters of television scan rates. SOLO1's principal logic blocks are:

- · CPU Bus Unit (busUnit)
- Main Memory Unit (memUnit)
- · RIO Bus Unit (rioUnit)
- Video Unit (vidUnit)
- Pixel Output Unit (potUnit)
- Graphics Engine (gfxUnit)
- Audio Unit (audUnit)
- I/O Device Unit (devUnit)
- Digital Video Input Unit (divUnit)
- Digital Video Encoder Unit (dveUnit)
- Video DAC Unit (dacUnit)
- Soft Modem Unit (modUnit)
- Smart Card UART (sucUnit)
- Miscellaneous/Clock Unit (mckUnit)
- · Other Units

A functional block diagram of the SOLO1 ASIC is shown in Figure 2-1.

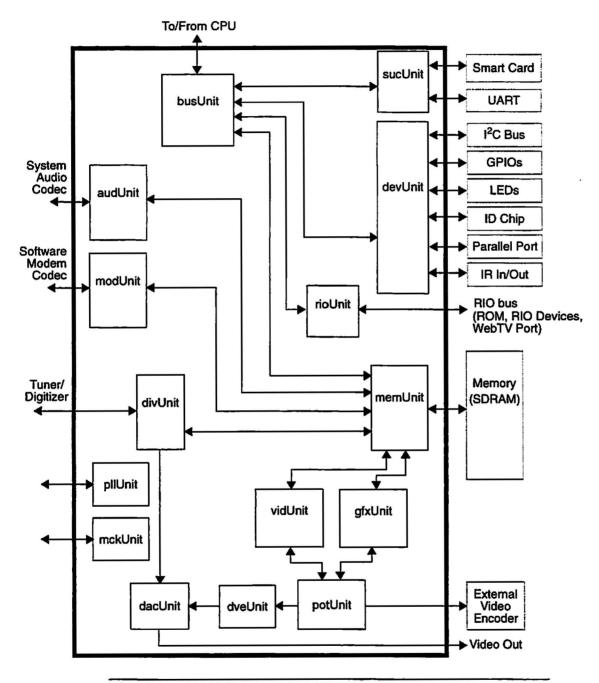


Figure 2-1 SOLO1 Functional Block Diagram

2.1 SOLO1 System Connections

Figure 2-2 shows the system connections and external components that can be used in a SOLO1-based system.

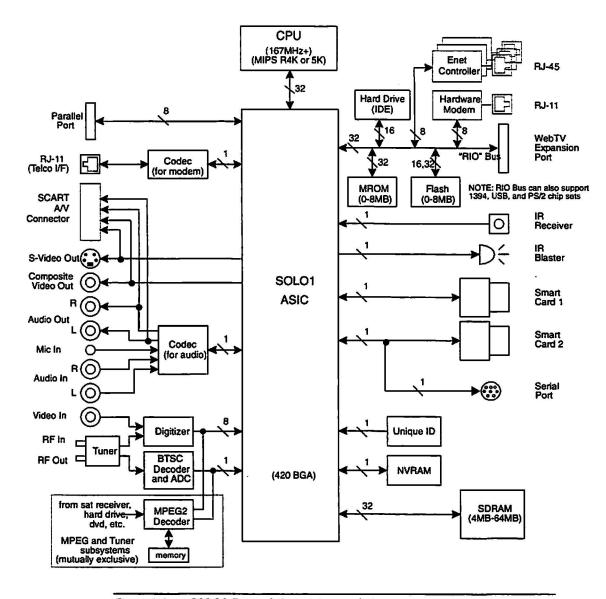


Figure 2-2 SOLO1 External Connections and Components

2.2 New Features

This section provides brief descriptions of the features that differentiate the SOLO1 ASIC from the earlier FIDO2 chip.

2.2.1 On-chip Video Encoder

SOLO1 provides an on-chip video encoder and on-chip DAC's. An external video encoder and DAC are also supported.

2.2.2 Digital Video Input Port

The Digital video/audio input ports allows SOLO1 to be used in a variety of systems that use satellite dish technology, digital tuners, or digital video disk players.

2.2.3 Internal Expansion Bus

SOLO1 provides for four internal expansion devices. The protocol of the expansion devices is ISA-compatible. One device may be the bus master, and the remaining three devices are slaves.

2.2.4 Dual Smart Card Interface

SOLO1 supports a second Smart Card interface. The intent of the second interface is to enable the use of a Security Access Module (SAM) in one interface, and the customer's card in the other interface. The SAM module provides a means of encrypting/unencrypting information contained on, or delivered to, the second Smart Card. The SAM module can also be used to provide conditional access to a satellite system.

2.2.5 Integrated Parallel Port and Serial Ports

SOLO1 provides a parallel port and a serial port, both of which require only external transceivers.

2.2.6 Soft Modem Support

SOLO1 has five signals that allow it to communicate with a modem CODEC. Software running on a SOLO1-based system can perform all modem signal processing required to produce up to a 56Kbit/sec modem, while significantly reducing system parts count.

2.2.7 IR Receiver and Blaster

SOLO1 incorporates IR receiver logic which previously required an external processor.

An IR Blaster is also supported, so that a SOLO1-based system can generate an IR signal.

2.2.8 Burst mode and Auto-Precharge mode on SDRAM

Burst/Auto-precharge use of SDRAM boosts SOLO1's memory performance to over 250Mbytes/sec (realizable), or 330Mbytes/sec (peak).

2.3 Functional Description

This section gives a functional description of each of SOLO1's principal logic blocks.

2.3.1 CPU Bus Unit

The busUnit supports the following CPUs:

- IDT R4640
- QED 5230
- Digital SA-110 (external glue logic required)

2.3.2 Memory Unit

The memory unit (memUnit) supports the following features:

- Up to 330 Mbytes/second peak bandwidth using SDRAM technology at 83.3MHz
- Support for 4 to 64Mbytes of SDRAM
- 32-byte Block Mode for all DMA devices improves throughput and efficiency
- Byte writes
- Interleave at the half-block boundary to hide pre-charge (auto precharge mode), and helps prevent starvation
- Random-access word mode supported for the graphics engine to accelerate rotations or odd memory strides.
- SDRAM power-down mode/auto-refresh supported.

The memUnit provides DMA interfaces for the following internal units:

- CPU (read/write)
- gfxUnit (YMaps, Cell Records, Pixels—read, Write-back Frame buffer—write)
- potUnit (Video Overlay Alpha channel —read)
- vidUnit (Frame Buffer—read)
- audUnit (audio buffer—read/write)
- modUnit (modern codec samples—read/write)
- divUnit (compressed MPEG—read, scaled video—write)

In addition to the interface logic, the memUnit also provides the memory refresh. Refresh has the highest priority of all memory accesses.

2.3.3 RIO Bus Unit

The RIO Bus Unit (rioUnit) provides SOLO1's interface to the RIO bus. The RIO bus provides the interface to ROM, and to both the on-board and off-board expansion devices.

The rioUnit supports up to 8Mbytes of Flash ROM and up to 8 Mbytes of Mask ROM, using two or four 16-bit wide parts. The data path to the external ROM is 32-bits wide.

The RIO bus supports up to four on-board expansion devices which are ISA-compatible, as well as the WebTV PortTM external expansion protocol.

2.3.4 Video Unit

The Video Unit (vidUnit) retrieves pixels from the frame buffer and passes them to the potUnit. The vidUnit must keep track of the following information to generate the proper frame buffer address and ensure the proper display of the pixels:

- Current Field in Frame
- Current Line in Field
- NTSC/PAL format

2.3.5 Pixel Output Unit

The Pixel Output unit (potUnit) runs at 2X the pixel clock. It receives pixels either from the vidUnit, which retrieves them from memory, or from the gfxUnit, which is producing them in real time, depending on whether or not the system is running in write-back mode or ping-pong mode.

2.3.6 Graphics Unit

The Graphics Unit (gfxUnit) accelerates the rendering of graphics in SOLO1-based systems. Rather than generating the entire pixel stream (as was the case with FIDO) software now only needs to convert high-level graphics image components into Y-Map entries and their associated cell-lists. The graphics engine supports the following features:

- Trapezoidal Cell Rendering
- X and Y Cell Scaling
- Cell Rotations
- Two types of pixel de-compression via Vector Quantization

The graphics unit can also deliver a pixel stream either directly to the potUnit for immediate display, or can write the stream to the frame buffer.

2.3.7 Audio Unit

The Audio Unit (audUnit) provides the interface between audio sample buffers in memory and the external DAC's and ADC's. The audUnit supports both system audio input and output.

2.3.8 I/O Device Unit

The I/O Device Unit (devUnit) provides a CPU interface to the following devices:

- IEEE 1284 Parallel Port (printer)
- · Infrared receiver and driver
- One I² C interface
- 16 CPU programmable General-Purpose I/O (GPIO) signals with interrupt capability
- LEDs

2.3.9 Digital Video Input Unit

The Digital Video Input Unit (divUnit) provides SOLO1 with the following features:

- Receives either 720 or 768 format digital video
- Scales digital video input to 1/4 screen or Half Screen Vertical or Half Screen Horizontal and DMA's the result to SDRAM Buffers.
- Maintains a separate DMA channel for vertical blanking information present in the digital video stream for interpretation by software (e.g., URLs, Closed-Captioning, SMPTE time codes).

2.3.10 Digital Video Encoder Unit

The Digital Video Encoder unit (dveUnit) performs the following functions:

- Receives either 720/768/640 pixel format streams from the divUnit or the potUnit, and generates Composite video, S-Video/Chroma and S-video/Luminance
- Optionally blends re-sampled graphics pixels (with Alpha appended) with the digital video stream to produce graphics overlays on the video.

2.3.11 Modem Unit

The Modem Unit (modUnit) provides the interface to an external codec for use with a software modem.

2.3.12 Smart Card and UART Control Unit

The Smart Card and UART Control Unit (sucUnit) contains the logic that links SOLO1 to the Smart Card and serial port interfaces. The serial port interface can be configured to be a second Smart Card interface, and supports a data rate of up to 115.2kbps.

2.3.13 Miscellaneous /Clock Unit

The Miscellaneous/Clock Unit (mckUnit) contains SOLO1's reset logic. Four types of reset are supported:

- Power-on
- Switching
- Watchdog
- Software

SOLO1 drives a system reset signal that is used to reset the other system components.

2.3.14 Other Units

There are five additional groups of support logic, which are described in the following sections.

2.3.14.1 Clock Driver Unit

The Clock Driver unit (ckdUnit) provides all of SOLO1's clocks.

2.3.14.2 Side Unit

The Side Unit (sidUnit) contains timing-critical logic that must be near the I/O pads.

2.3.14.3 I/O Unit

The I/O Unit (ioUnit) contains the I/O pad, drivers and receivers.

2.3.14.4 PLL Unit

The PLL Unit (pllUnit) contains the phase-locked loop and its associated control logic.

2.3.14.5 DAC Unit

The DAC Unit (dacUnit) contains the three digital-to-analog convertors used by the dveUnit.

gfxUnit Theory of Operations

This chapter provides an overview of how SOLO1's Graphics Engine (gfxUnit) works, as well as some general background information about how images are rendered.

3.1 gfxUnit Overview

The gfxUnit's operation is based on two key features:

- Square pixels are used for both NTSC and PAL.
- 2. SOLO1's image data is in YCbCr format, the native format of NTSC, PAL, JPEG, and MPEG. Notably, SOLO1 does not support RGB, the native format of Macintosh, PC, and workstations. YCbCr was chosen because it models the characteristics of the human visual system better than RGB, and as a result requires less data for a given level of image quality. Y describes the perceptual brightness of a pixel and Cb and Cr describe the color characteristics of a pixel. Since the acuity of the human visual system is better in regard to brightness than it is in regard to color, it is possible to represent Cb and Cr at lower resolution than Y without perceptual loss in image quality. SOLO1 takes advantage of this characteristic, allowing the CbCr data to be stored at a lower resolution than the Y data.

3.2 Object-oriented Graphics

The gfxUnit uses an object-oriented approach to graphics. Examples of graphics objects might be: a JPEG image, a block of text, or a graphical element such as the selection bar. The software prepares a list of these objects including each object's location in memory and its position on the screen. Also included in the object description are parameters that control the filtering (for flicker reduction), blending (for special effects), scaling, rotation, or compression.

The gfxUnit is a display list-based compositing engine, sourcing various textures from RAM and compositing them together in the on-chip Scanline buffer. The contents of the Scanline buffer can then be displayed directly to the television display screen, or written back to a frame buffer in memory.

3.2.1 Compositing

All writes to the on-chip Scanline Buffer are actually Alpha blends with the foreground pixel being the new pixel being written and the background pixel being the pixel already in the Scanline buffer.

The read-modify-write operates as follows The existing pixel value is read, it is combined arithmetically with incoming pixel value, and then the result is written back to the Scanline buffer. The operations are pipelined so that there is a pixel written each cycle.

3.2.2 Cels

A Cel is a data structure in WebTV memory. It defines the mapping of a twodimensional image, stored in memory, to a region in an interlaced video frame. We use the term "Cel" since these 2D object serve a similar function as Cels used by traditional animators. Examples of various Cels are shown in Figure 3-1.

One or more Cels are combined to form the final video output display.

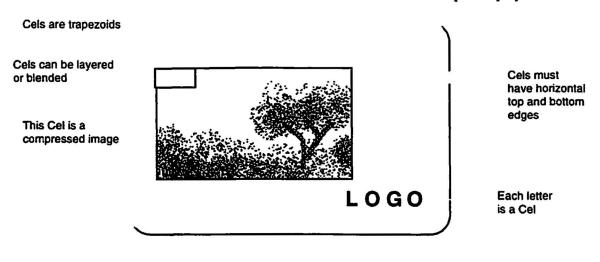


Figure 3-1 Cel Types

As the gfxUnit generates each line of the video image, it reads in the parts of the active Cels that intersect that line and composites them together. The resulting video frame, generated line-by-line, is a composite image of all of the Cels that the gfxUnit has loaded from memory.

A Cel is composed of a CelRecord (which describes the Cel's characteristics), a TextureMap (which is a 2D array of up to 256x256 4-, 8-, 16-, or 32-bit pixels), and optionally a Codebook (which is used in compressed Cel formats). TextureMaps are used for displaying images and anti-aliased text from either full resolution or compressed source data. Spatial transforms can be applied to all TextureMaps, whether stored in compressed form or not.

Cels may source their image data from a variety of structures in memory, both compressed and uncompressed. Cels may be opaque or translucent, and they may be as small as a single pixel or as large as the whole screen.

A Cel's displayed region on the screen is defined by a top line, a bottom line, a left edge, and a right edge. The top line and bottom line define the scanline at which the Cel begins and the Cel ends on the screen. The left and right edges start at subpixel positions on the top line, and then each has a slope which defines an angle to the bottom line. Thus, the displayed region of each Cel on the video frame is a scanline-aligned trapezoid. Note that rectangles and triangles with one edge aligned to a scanline are degenerate cases of scanline-aligned trapezoids.

Several example Cels are shown in Figure 3-2. Cel 0 is occupies the entire screen space. Cel 1 is a rectangular Cel. Cel 2 is a triangular Cel. Cel 3 is a trapezoidal Cel. Cels 4, 5, and 6 are concatenated together to form a larger polygon. Note that any quadrilateral smaller than 256x256 pixels can be represented by three screen-aligned trapezoids.

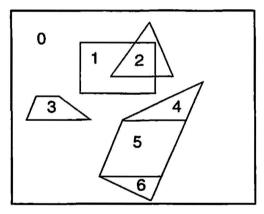


Figure 3-2 Example Cels

The Cels structure is a list of all CelRecords, packed one after another, aligned to 8-byte (64-bit) boundaries. Cels can be aggregated into CelBlocks. CelBlocks are groups of Cels which are displayed from the same yTop to yBottom on the screen. For example, a typical CelBlock is a line of text on the screen in which each text character is a separate Cel.

3.2.3 yMap Structure

The yMap structure specifies the top and bottom line of each CelBlock to be displayed on the screen. Each 32-bit word in the yMap is a pointer to a CelBlock. A CelBlock which is not pointed to by a yMap entry will not be displayed. The last yMap entry is flagged by a termination value.

The yMap structure specifies which CelBlocks are visible on each active line and the back-to-front order of each of the CelBlocks on the screen. The gfxUnit scans through the entire yMap on each scanline, and upon consideration of each yMap entry, is directed to load or not load each CelBlock.

Each entry in the yMap defines the top and bottom line (inclusive) of each CelBlock in the displayed image. By comparing these top and bottom parameters with the current scanline, the gfxUnit is able to determine if a given CelBlock is to be loaded. Since the gfxUnit reads the yMap from beginning to end on each scanline, the order of CelBlocks in the yMap determines the load order of CelBlocks into the gfxUnit, and hence, the yMap entry ordering determines the back-to-front ordering of CelBlocks on a scanline. Within a CelBlock, the Cels are loaded in the order that they appear in the Cels structure.

The first entry in the yMap is the background-most CelBlock, each succeeding entry is increasingly closer to the foreground, until the last entry in the yMap is the foreground-most CelBlock.

Normally, there is a single yMap for the entire image, but it is possible to change yMaps during active video to accommodate very complex images.

3.2.4 Cels Structure

The CelRecords of all Cels to be displayed in a given frame are stored in a Cels Structure. CelRecords in the same CelBlock are grouped together. CelBlocks are stored in the Cels list in any order, but Cels within a CelBlock are stored in back-to-front order.

Each Cel has a CelRecord which includes the parameters (other than the parameters in its yMap entry) that describe the characteristics of the Cel. These parameters include:

- the location of the texture in memory
- the width of the texture
- the location of the optional codebook in memory
- position of the Cel on the screen
- blending characteristics
- scaling parameters
- rotation-parameters

- · slope of the right and left edges of the Cel
- whether the texture is to be repeated

3.3 TextureMaps

A TextureMap may be stored anywhere in memory, including in ROM. There are different types of TextureMaps supported by the gfxUnit, each providing different trade-offs in terms of memory requirements and display quality.

A Cel indexes individual pixels from a TextureMap using the u and v variables, which are both 8.8 values. This limits the size of a cel's TextureMap to 256 pixels by 256 pixels. However, TextureMaps larger than this can be used if they are broken up into separate Cels, with each Cel indexing up to 256 pixels in each of the u and v directions.

u=0 and v=0 is the center of the center pixel of the TextureMap, and u and v each increase by 1 for each subsequent pixel horizontally and vertically, respectively.

The gfxUnit supports four TextureMap formats:

Direct

32 or 16 bits/pixel

Vector Quantization

8 or 4 bits/pixel

3.3.1 Direct TextureMap Formats

The two Direct TextureMap formats correspond to the two pixel specification formats:

4:4:4

YCbACr 32 bits/pixel

4:2:2

YCbYCr 16 bits/pixel

These formats are quite expensive in terms of memory consumption, but they permit the most flexibility. Their primary applications are for small objects and for supporting data that is input from external devices in raw form (e.g. a video digitizer would probably capture data in 4:2:2 format).

Figure 3-3 shows the organization of 6x7-pixel 4:4:4 Direct TextureMap (note that only positive u, v values are used in this example).

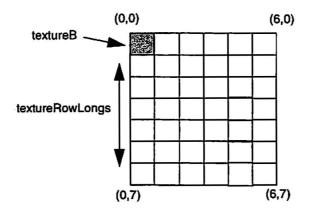


Figure 3-3 6x7-pixel 4:4:4 Direct TextureMap

A TextureMap is stored in successive memory locations starting at textureBase (which falls on an even long (32-bit) word boundary). The gray box indicates the area covered by one 32-bit word (1 pixel) in a 4:4:4 Direct TextureMap. The distance (in 32-bit words) from one pixel to the pixel directly below it is specified by textureRowLongs.

Figure 3-4 shows the organization of 4x7-pixel 4:2:2 Direct TextureMap (note that only positive u, v values are used in this example).

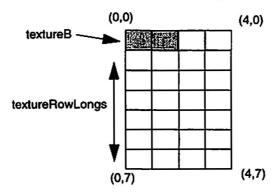


Figure 3-4 12x7-pixel 4:2:2 Direct TextureMap Organization

The gray box indicates the area covered by one 32-bit word (2 pixels) in a 4:2:2 Direct TextureMap. Note that the left-most pixel in the YCbYCr pixel corresponds to the (0,0) coordinate.

3.4 VQ TextureMap and Codebook Formats

VQ (Vector Quantization) is a simple compressed image format that provides fairly high quality with full generality of spatial manipulation, and very simple (single look-up) decompression.

To understand how Vector Quantization works, consider a simple form of VQ in use today: the frame buffer with a Color Look-Up Table (CLUT).

For reasons of RAM limitations or CPU throughput constraints, computers often have an 8-bit/pixel frame buffer rather than a 24-bit "true color" frame buffer. Such a low-bit-depth frame buffer can only display 256 colors at a time on the screen. Although 256 colors are far fewer than the 16 million colors a "true color" frame buffer is capable of displaying at once, a 256-entry CLUT on the back-end of the frame buffer allows the application to specify each of these 256 colors to be one of 16 million, and consequently the CLUT can be tailored to accommodate the color requirements of a given image.

For example, if a sunset image is being displayed, the CLUT can be loaded with reds. If a frog image is being displayed, the CLUT can be loaded with greens. But of course, the 256 colors must be shared among all the pixels displayed, and if there is a great deal of color variation in the image displayed, there may not be enough unique colors to go around and the image may not be rendered very realistically.

Or, put another way, on a 640x480 monitor there are some 300,000 pixels. If there are only 256 unique colors for these 300,000 pixels, then on average, each of the 256 colors must be used more than a thousand times. Needless to say, choosing that list of 256 colors and doling them out to the 300,000 pixels—to provide optimal image quality—is not only tricky business, it's an art form.

And your reward for going to all of this trouble of distributing 256 colors among 300,000 pixels is a measly 3:1 compression ratio. That is to say, you've taken 24-bit pixels and represented them as 8-bit pixels, and therefore you realize a 3:1 reduction in RAM. Not very exciting, given the constraints you have to deal with in an 8-bit color space, but nonetheless, we have demonstrated an important principle: by allocating a small amount of memory for a CLUT (in this case 256 \times 24-bits, or 768 bytes) we are able to reduce the memory requirements of an image (albeit with some loss in image quality).

Formally, VQ is a method whereby an image is "quantized" into "vectors" which then point to entries in a "codebook". In the case of an 8-bit frame buffer with a CLUT, each 8-bit value in the frame buffer is a "vector" and the CLUT is the "codebook". And when we perform this art form of choosing our 256 colors and distributing them to the 300,000 pixels of the image, what we are doing is "quantizing" the image.

Of course, the 3:1 compression ratio of the 8-bit frame buffer implementation of VQ is nothing to write home about, but there is another wrinkle to a general implementation VQ that makes it much more interesting: each vector can actually represent many pixels in the uncompressed image instead of just a single pixel.

For example, in the frame buffer with CLUT example, each vector in the frame buffer corresponded to a single pixel in the uncompressed image and pointed to a single color in the codebook (i.e. the CLUT). But imagine that each vector in the frame buffer corresponded to a 2x2 block of pixels in the uncompressed image and pointed to a 2x2 block of colors in the codebook. Of course, that art form of choosing 256 entries for the codebook and distributing amongst the pixels of the original image becomes even more difficult since we're now distributing 2x2 blocks of pixels rather than single pixels (i.e. 1x1 blocks), but the reward is a significantly higher compression ratio. Specifically, since each vector now represents 4 pixels of the original image, we would achieve an additional 4:1 reduction in RAM requirements, for a total of 12:1 compression over the original image.

So, now we can consider VQ in its more general form. A VQ vector corresponds to a to a block of pixels in the uncompressed image. The codebook entries are the same size as the blocks pointed to by the vectors. So, for example, if we have 8-bit vectors and a 2x2 block size, there will be 256 2x2 blocks in the codebook.

Now, there are several points worth noting about VQ. First of all, VQ is a "lossy" compression technique which does a better or worse job compressing an image depending on the coherence of the image (i.e. how self-similar the image is), and how big the image is relative to the size of the codebook. In other words, if the image is a sunset with all reds, it may look better compressed with VQ than an image of a bowl of fruit with a wide variety of colors. Also, an image with 50,000 pixels will probably look better than an image with 300,000 pixels, given the same size codebook, for the simple reason the limited number of colors in the codebook are shared amongst fewer pixels. Of course, there are other factors that figure into the quality of a compressed image, but the key point is that VQ compression will give you varying results depending on the image material and the size of the image relative to the codebook.

Additionally, it is important to note that unlike conventional graphics systems where there is a single CLUT (i.e. codebook) for the entire frame buffer (e.g. making it difficult to display both a sunset and a frog at the same time because they share the same 256 colors), the gfxUnit supports a different codebook for each VQ image. So, for example, if you intend to display both a sunset and a frog at the same time using the gfxUnit, you'd probably allocate a separate codebook for each.

Finally, the gfxUnit's approach to VQ is a little different than textbook VQ in that it supports an 8-bit Alpha channel. So, if you have an image with translucency, or perhaps curved edges, you can compress the translucent pixels (including antialiased edge pixels) along with the rest of the pixels.

3.4.1 VQ Format Summary

The gfxUnit supports two VQ formats: 4:4:4 and 4:2:2; and two vector sizes: 8-bit and 4-bit. 4:2:2 is only supported in 8-bit vector mode. Thus, there are three supported VQ modes (see Table 3-1).

Table 3-1 gfxunit Formats

Vector Size	Pixel Format	Block Size	Compression Ratio
8 bit	4:4:4	1x1	3:1
4 bit	4:4:4	1x1	6:1
8 bit	4:2:2	2x2	12:1

3.4.1.1 VQ/4:4:4

VQ/4:4:4 is a 1-pixel tall version of VQ (see Figure 3-5).

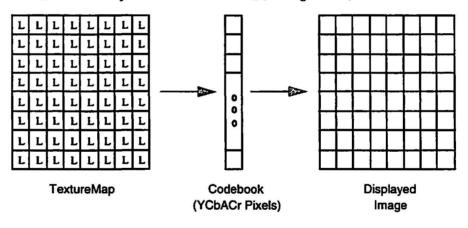


Figure 3-5 VQ/4:4:4 Format

The TextureMap has a vector (designated by "L" for "Lookup index") at each pixel location. This vector is either 4 bits or 8 bits in size. For each pixel displayed in the 4:4:4 image, the gfxUnit fetches a single L. The gfxUnit then uses L to index into the Codebook, and then the YCbCr pixel fetched from the Codebook is used for display (i.e. is composited into the Scanline buffer).

For each pixel composited into the scanline buffer, there are two fetches from memory: first a vector (L) is fetched, then a YCbCr pixel from the Codebook is fetched.

An on-chip Codebook RAM has been included in the gfxUnit for VQ4 4:4:4 mode. Once the Codebook has been loaded, only one fetch from memory is required for the 4-bit vector per pixel.

For Cels with an off-chip Codebook, first the gfxUnit fetches 16 vectors and stores them in a vector cache. Next, the gfxUnit fetches the Codebook pixels pointed to by the vectors in the vector cache and stores them in the Scanline buffer. Once the vector cache is exhausted, the gfxUnit fills it up again, and then goes back to loading Codebook pixels. This continues back-and-forth until a line of the TextureMap has been completely loaded.

There are several points worth noting:

- Rather than fetching data in the sequence vector-codebook-vector-codebook...etc., the gfxUnit fetches data in the sequence vector-vector-vector-vector-vector-codebook-codebook-codebook, etc. This data is fetched in this order to optimize accesses to SGRAM since the vectors are likely to be on the same page of SGRAM, and the codebook should be on the same page of SGRAM.
- 2. Although Codebooks are defined to be either 16 or 256 entries long, it is quite possible to have smaller Codebooks by simply not using the higher vector numbers. For example, a 64-entry Codebook could be used with an 8-bit vector, so long as you make sure the vector numbers stay below 64. This can be used to minimize memory consumption for small objects such as icons.
- 3. Large TextureMaps can be subdivided into multiple adjacent Cels, and each Cel can refer to its own Codebook. Although this is clearly less efficient from a memory consumption point of view, it will result in higher image quality. Also, the smaller the TextureMap, the faster a compression algorithm can generate a given quality Codebook.
- 4. Since there is an 8-bit Alpha value associated with each Codebook entry, images can be compressed with translucency per pixel. This is very useful for anti-aliasing edges of the image.
- With 4-bit vectors the Codebook requires 16 entries of 32 bits each, or 64 bytes. With 8-bit vectors the Codebook requires 256 entries of 32 bits each, or 2K bytes.

3.4.1.2 VQ/4:2:2

The gfxUnit VQ/4:2:2 is a two-pixel tall version of VQ (see Figure 3-6).

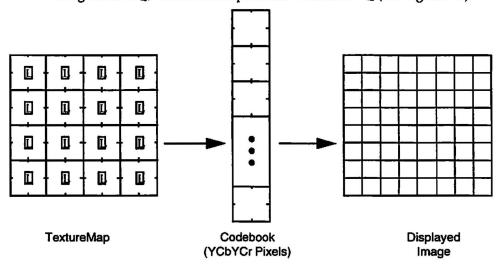


Figure 3-6 VQ/4:2:2 Format

Note that unlike VQ/4:4:4 pixels, in this format each vector L corresponds to four pixels in the source image. This follows naturally from the fact that because there are two YCbYCr pixels in each Codebook entry, each Codebook lookup yields two pairs of pixels.

All of the points noted in regard to VQ1 with 4:4:4 apply to this format as well, except:

- 1. Since one-fourth as many vectors are needed to represent the source image, an additional 4:1 compression ratio is achieved.
- 2. The Codebook is twice as large since each vector now points to two 32-bit words instead of one.
- Because 4:2:2 format is utilized in this Codebook, there is no Alpha value stored per pixel, other than transparency if a Y value is 0.

3.5 Forward Texture Mapping

SOLO1 employs "forward texture mapping" to implement its texture-to-screen mapping. That is to say, as the gfxUnit scans through x, y space it calculates a u, v for each x, y; it finds the closest pixel in the TextureMap that corresponds to the computed u, v; and then it reads that pixel and stores it at the x, y location.

Because SOLO1 is a Scanline buffer architecture, it considers all the x's at a given y before advancing down to the next y. As such, all of the math for texture-to-screen space mapping is organized to be incremental in x. The following algorithm illustrates forward texture mapping that is incremental in x. This algorithm is a simplification of that used by the gfxUnit, but it is instructive since it is easy to understand.

```
int u, v;  // horizontal and vertical TextureMap coords
int x, y;  // horizontal and vertical screen coordinates
int dux, dvx; // change in u, v for each increment in x
int duy, dvy; // change in u, v for each increment in y

for (y = topLine; y <= bottomLine; y++) {
    u += duy; v += dvy;

    for (x = xLeft; x <= xRight, x++) {
        u += dux; v += dvx;
        screen(x,y) = TextureMap(u/v);
    }
}</pre>
```

Note that the above algorithm assumes the destination is a rectangle bounded by topLine, bottomLine, xLeft, and xRight.

Video Tuner/Digitizer Theory of Operations

4.1 Overview

SOLO1's tuner/digitizer logic supports the following features:

- Cable and broadcast television tuning, with stereo audio
- Video and graphics integration, from simple picture-in-picture (PIP) to overlay
- No video quality compromise when watching full-screen TV without overlay
- · No screen roll when turning overlay on or off

4.2 Design Issues and LC2 Implementation

Here are the design issues that effected the tuner/digitizer architecture:

 Video quality is degraded when the signal is decoded and re-encoded before going to the TV.

If video is always decoded from the tuner and passed through the system, there will be a noticeable increase in the color separation artifacts. A simple solution to this problem is to bypass the decode/encode path when the user is watching full-screen video. This can be accomplished in two ways: a passive bypass circuit between the tuner and the RF modulator, and an active bypass in the digital domain from the ADC in the video decoder to the DACs in the video encoder.

The first of these, RF bypass, works fine when the user primarily connects his/her TV to the WebTV device through the RF modulator. But if composite baseband video is used, it will be necessary to perform the active bypass. Furthermore, if the WebTV device is connected to the TV via S-video, then bypass is not an option, and there must be a color separation step somewhere.

The solution to this problem is to detect the connection of the S-video port, and if connected, disable the bypass. Note that, while the active bypass works for both composite and RF, only the RF bypass works when the user unplugs the WebTV device, so RF bypass should also be made available.

2. Synchronization between the input and output.

We could implement genlock and keep output and input always in sync, but that raises other problems when the input becomes unstable (e.g. during a channel change).

A better solution is to implement a time-base corrector, which means the video data is passed through memory, so the output and input can run independently. In this way, the system is insulated from glitches in the input video and always produces a stable raster. This also lets you to do interesting things: like transition effects on channel changes.

The system runs as a time-base corrector when not in bypass, but when there is a transition from bypass to video/graphics overlay, the screen rolls because the output signal wasn't at all in sync with the input. This will happen every time that the overlay is brought up, and again when it is taken away.

Screen roll can be eliminated by SOLO1's internal video encoder. This video encoder can make both coarse and fine adjustments in the video output timing. The encoder can drop a field, drop a line from the bottom, or eat a few clocks from the last line.

By applying these controls, the output video can be kept near enough in sync with the input to prevent screen roll. However, to do this, you must be able to measure the phase difference between the output and the input. The DIVIT port, in SOLO1, keeps track of the time between its odd-field start and the odd-field start of the output. This difference can be monitored and made to match that of the system in bypass.

The following table illustrates the major modes of operation of the tuner/digitizer subsystem.

Table 4-1 Tuner/Digitizer Operating Modes

Signal Type	Power Off	Power On/ TV mode	Overlay
S-Video	no video	decode/encode	decode/encode
Composite	no video	active bypass	decode/encode
RF	RF bypass	active bypass	decode/encode

Figure 4-1 shows the LC2 logic that is involved in creating the tuner/digitizer solution. The three logic blocks outlined in bold represent the Digital Video Interface Unit (divUnit), the Digital Video Encoder (dveUnit), and the Tuner. These units are described in more detail in the following sections.

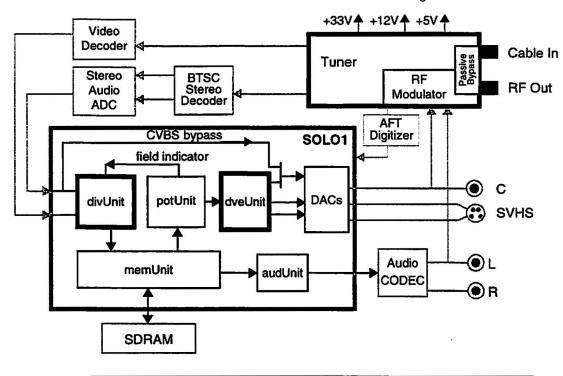


Figure 4-1 Tuner/Digitizer and Supporting Logic

4.2.1 Digital Video Interface Unit (divUnit)

The divUnit port is a basic video DMA channel that captures video fields and dumps them into memory. This port also has the ability to capture data from the vertical blanking interval (VBI) and direct it to a separate DMA channel from the video being captured. The control registers describe a rectangle around the data in each video field and the VBI to be captured. The entire set of DMA registers is shadowed to allow the next capture to be programmed while the first capture is in progress. The shadow set can also operate in a ping-pong mode with the active set to provide a continuous two-field DMA operation.

To support non-square-pixel sources, like a CCIR601 standard decoder, the divUnit contains a resampler to convert CCIR601 aspect ratios to square pixels. The modes supported are 720->640 (NTSC), 720->768 (PAL), 720->320, 720->384, 1->1, and 2->1.

Video can also be decimated 2->1 or 1->1 prior to writing it to memory. This could be useful for creating a PIP-style video window, without burdening the memory subsystem.

The divUnit is designed to accept data from three different interfaces/protocols: explicit sync signal pins; an 8-bit CCIR656 (D1) interface; and the Brooktree 8-bit "ByteStream" protocol. This flexibility allows SOLO1 to support a broad range of MPEG decoders and analog video decoders.

Finally, DIVIT has an input-only, three-pin PCM audio input, designed to accept uncompressed audio from either an analog codec or an MPEG audio decoder.

4.2.2 Digital Video Encoder (dveUnit)

The DAC outputs are paired as follows: Chroma/Blue, Composite/Red, and Luma/Green. The output impedance of the DAC is 100 Ohms. Output voltages range from VDD to VDD - 1. The maximum output swing is 1Volt peak-to-peak.

The DVE provides two major output modes — Full Range (FR) and Fixed IRE (FI). In FR mode, the output signals occupy (nearly) the entire range of DAC outputs from 0 to 1023. For composite video, the required gain on the Y and C DACs is essentially unity; however, the composite DAC requires a gain factor of 1.32 in order to achieve the correct range of signal outputs. RGB outputs are also nearly at unity gain in FR mode. The outputs may also be programmed for positive or negative (inverted) polarity.

Table 4-2 details the output levels for each of the six DVE modes. The "Codes/IRE" column represents the increase in the digital value at the DAC digital inputs required to create a single IRE at the output. "Full Scale" is assumed to be 1Volt in all cases (i.e. the difference between the output at 1023 and 0). "Max Signal" is the largest peak-to-peak signal that will be observed for any video signal. "Amplification Factor" is the amount by which the output must be multiplied to achieve the proper output voltage range.

Normally, the DVE will be in YC_FR mode. RGB_FR will support RGB applications. Other modes may be programmed if useful.

Output Mode	· I DAL IMP		Full Scale (V)	Max Signal (V)	Amplifica- tion Factor (Vo/Vi)
YC_FR	VDAC	2.7675	1.00	0.92	1.32
	YDAC	3.53625	1.00	0.97	1.03
	CDAC	3.57611	1.00	0.89	1.02
	VDAC	2.7675	1.00	0.92	1.32

Table 4-2 DVE Output Mode Signal Levels

Table 4-2 DVE Output Mode Signal Levels (Continued)

Output Mode	DAC	Codes / IRE (IRE-1)	Full Scale (V)	Max Signal (V)	Amplifica- tion Factor (Vo/Vi)
YC_FI	YDAC	2.7675	1.00	0.76	1.32
•	CDAC	2.7675	1.00	0.68	1.32
	RDAC	3.513125	1.00	0.69	1.04
RGB_FR	GDAC	3.513125	1.00	0.96	1.04
	BDAC	3.513125	1.00	0.69	1.04
	RDAC	2.7675	1.00	0.54	1.32
RGB_FI	GDAC	2.7675	1.00	0.76	1.32
	BDAC	2.7675	1.00	0.54	1.32
	RDAC	5.11	1.00	1.00	1.00
BYP_FR	GDAC	5.11	1.00	1.00	1.00
	BDAC	5.11	1.00	1.00	1.00
	RDAC	2.7675	1.00	0.54	1.32
BYP_FI	GDAC	2.7675	1.00	0.54	1.32
	BDAC	2.7675	1.00	0.54	1.32

4.2.2.1 Sample Applications

4.2.2.1.1 Simple S-video/Composite Output — Discrete The following block diagram shows one possible method of implementing simultaneous Y/C/Composite outputs. This is the preferred implementation when the amplifiers are constructed from discrete transistors. Inverted outputs are used because inverting amps may be made from fewer transistors.

The output swing is amplified to twice the desired value to allow for the attenuation from the series termination resistor.

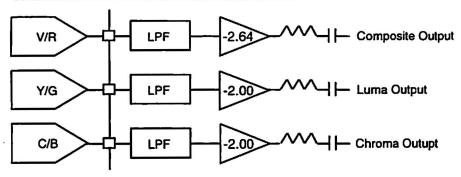


Figure 4-2 Simultaneous Y/C/Composite Outputs

4.2.2.1.2 Simple S-video/Composite Output — Integrated Output Buffer Several vendors make analog output ICs intended for use with digital video encoders. MicroLinear makes a chip that integrates all of the functions within the shaded area.

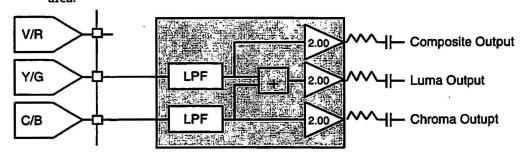


Figure 4-3 Analog Output IC Functions

4.2.2.1.3 SCART Connector with Composite, Y/C, and RGB RGB and composite/Y/C outputs may be supported (although not simultaneously). The following diagram shows how this is achieved through the addition of one more LPF and three more amplifiers to the previous configuration. In this case, the circuit could be implemented with a three-channel amplifier IC, plus the LPF/amp/adder as before (i.e. two ICs). Each component is buffered separately in order to avoid problems with double termination.

If the SYNCBYPASS bit, in the DVE_CNFG register, is set to '1', all of the enbedded syncs are removed, both in RGB and in YUV modes.

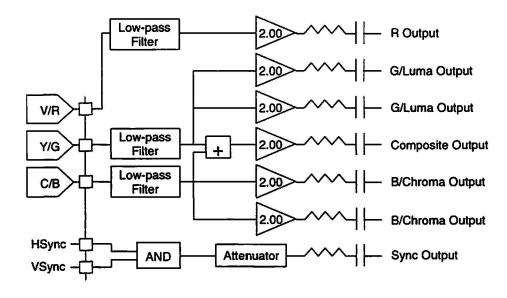


Figure 4-4 SCART Connector with Composite, Y/C and RGB

4.2.2.2 VGA Timing

The sync output pin shown in Figure 4-4 is actually an I/O pin. When the DVE is in slave mode (and the pin functions as an input), VGA timing signals may be driven in on the HSYNC and VSYNC inputs.

4.3 Tuner

The tuner must have the following minimum set of features:

- BTSC/MTS Stereo audio (internal to tuner preferable, external chip more likely)
- SAP (Second Audio Program) option
- Baseband video output
- Compatibility with US Cable TV systems (a global version would be preferable)

Preferred, but less important:

- I2C interface control useful, but not more important than cost
- RF pass-through or RF modulator built in, or both. (Ideally, we need the kind
 of functionality found in VCRs: tuning, RF output from internal source, and
 RF pass-through when the device is off.)

Software Description

This chapter provides descriptions of all of the SOLO1 registers.

5.1 Memory Map

Table 5-133 outlines the memory map of the WebTV™ system.

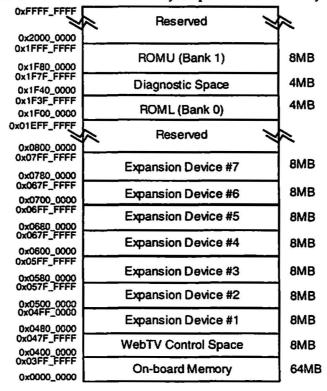


Figure 5-1 WebTV™ Memory Map

5.2 Registers

5.2.1 Register Summary

The address, register name, and a short description of each register in the SOLO1 control space is in Table 5-1. The 3-letter prefix before the register name indicates where, within SOLO1, the register physically resides. The registers can be referred to with or without these 3-letter prefixes. More detailed descriptions of individual registers are found later in this chapter.

Note: All registers are accessible only as 32-bit aligned words.

Table 5-1 SOLO1 Registers - Arranged by Address

Address	Register Name	R/W	Description
busUnit Regist	ers		
0x04000000	BUS_CHIPID	R/W	SOLO1 chip ID
0x04000004	BUS_CHPCNTL	R/W	SOLO1 chip control
0x04000008	BUS_INTSTAT	R/W	Interrupt status
0x04000050	BUS_INTSTAT	Set	Interrupt status - set
0x04000108	BUS_INTSTAT	Clr	Interrupt status - clear
0x0400000c	BUS_INTEN	R/Set	Interrupt enable - set
0x0400010c	BUS_INTEN	Clr	Interrupt enable - clear
0x04000014	BUS_ERREN	R/Set	Error enable - set
0x04000114	BUS_ERREN	Clr	Error enable - clear
0x04000018	BUS_ERRADDR	R/W	Error address
0x0400001tc 0x0400002c	RESERVED	N/A	
0x04000030	BUS_WDVALUE	R/W	Watchdog counter reset value
0x04000034	BUS_LOWRDADDR	R/W	Low memory read protection address
0x04000038	BUS_LOWRDMASK	R/W	Low memory read protection mask
0x0400003c	BUS_LOWWRADDR	R/W	Low memory write protection address
0x04000040	BUS_LOWWRMASK	R/W	Low memory write protection mask
0x04000044	KESBRVED	N/A	医神道神经
0x04000048	BUS_TCOUNT	R/W	Timer Counter
0x0400004c	BUS_TCOMPARE	R/W	Timer Compare
0x04000054	BUS_ERRSTAT	R/Set	Error status - set

 Table 5-1
 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x04000010	BUS_ERRSTAT	W	Error status
0x04000110	BUS_ERRSTAT	Clr	Error status - clear
0x0400005c	BUS_GPINTEN	R/Set	GPIO interrupt enable - set
0x0400015c	BUS_GPINTEN	Clr	GPIO interrupt enable - clear
0x04000060	BUS_GPINTSTAT	R/Set	GPIO interrupt status - set
0x04000058	BUS_GPINTSTAT	W	GPIO interrupt status
0x04000158	BUS_GPINTSTAT	Clr	GPIO interrupt status - clear
0x04000064	BUS_GPINTPOL	R/W	GPIO interrupt polling
0x0400006c	BUS_AUDINTSTAT	R/Set	audUnit interrupt status - set
0x04000068	BUS_AUDINTSTAT	W	audUnit interrupt status
0x04000168	BUS_AUDINTSTAT	Clr	audUnit interrupt status - clear
0x04000070	BUS_AUDINTEN	R/Set	audUnit interrupt enable - set
0x04000170	BUS_AUDINTEN	Clr	audUnit interrupt enable - clear
0x04000078	BUS_DEVINTSTAT	R/Set	devUnit interrupt status - set
0x04000074	BUS_DEVINTSTAT	W	devUnit interrupt status
0x04000174	BUS_DEVINTSTAT	Clr	devUnit interrupt status - clear
0x0400007c	BUS_DEVINTEN	R/Set	devUnit interrupt enable - set
0x0400017c	BUS_DEVINTEN	Clr	devUnit interrupt enable - clear
0x04000084	BUS_VIDINTSTAT	R/Set	vidUnit interrupt status - set
0x04000080	BUS_VIDINTSTAT	w	vidUnit interrupt status
0x04000180	BUS_VIDINTSTAT	Clr	vidUnit interrupt status - clear
0x04000088	BUS_VIDINTEN	R/Set	vidUnit interrupt enable - set
0x04000188	BUS_VIDINTEN	Clr	vidUnit interrupt enable - clear
0x04000090	BUS_RIOINTSTAT	R/Set	RIO bus interrupt status - set
0x0400008c	BUS_RIOINTSTAT	W	RIO bus interrupt status
0x0400018c	BUS_RIOINTSTAT	Clr	RIO bus interrupt status - clear
0x04000094	BUS_RIOINTPOL	R/W	RIO bus interrupt polling
0x04000098	BUS_RIOINTEN	R/Set	RIO bus interrupt polling - set
0x04000198	BUS_RIOINTEN	Clr	RIO bus interrupt polling - clear
0x040000a0	BUS_TIMINTSTAT	R/Set	Timing interrupt status - set
0x0400009c	BUS_TIMINTSTAT	W	Timing interrupt status

 Table 5-1
 SOLO1 Registers - Arranged by Address (Continued)

0x0400019c BUS_TIMINTSTAT Clr Timing interrupt status - clear 0x040000a4 BUS_TIMINTEN R/Set Timing interrupt enable - set 0x040000a8 RESETCAUSE R/Set Reset cause - set 0x040000a0 RESETCAUSE Clr Reset cause - clear 0x040000b0 BUS_JIFENLADDR R/W Lower bound of Java1 R/W protection fence 0x040000b4 BUS_JIFENHADDR R/W Upper bound of Java2 R/W protection fence 0x040000b6 BUS_J2FENLADDR R/W Lower bound of Java2 R/W protection fence 0x040000bc BUS_J2FENHADDR R/W Upper bound of Java2 R/W protection fence 0x040000c0 BUS_J2FENCECNTL R/W Size of the physical RAM 0x040000c0 BUS_FENCECNTL R/W Cancels writes to the J1 and J2 valid address ranges 0x040000c0 BUS_BOOTMODE R/W CPU reset string 0x040000c0 BUS_USEBOOTMODE R/W CPU reset string 0x04001000 RIO_SYSCONFIG R/W HW/SW system configuration 0x04001000 ROM_CNTL1 R/W ROM control for Bank 1	Address	Register Name	R/W	Description	
0x040001a4 BUS_TIMINTEN Clr Timing interrupt enable - clear 0x040000a8 RESETCAUSE R/Set Reset cause - set 0x040000b0 BUS_JIFENLADDR R/W Lower bound of Java1 R/W protection fence 0x040000b4 BUS_JIFENHADDR R/W Upper bound of Java2 R/W protection fence 0x040000b6 BUS_JZFENLADDR R/W Lower bound of Java2 R/W protection fence 0x040000bc BUS_JZFENHADDR R/W Upper bound of Java2 R/W protection fence 0x040000c0 BUS_JZFENHADDR R/W Upper bound of Java2 R/W protection fence 0x040000c0 BUS_JZFENHADDR R/W Upper bound of Java2 R/W protection fence 0x040000c0 BUS_JZFENHADDR R/W Upper bound of Java2 R/W protection fence 0x040000c0 BUS_JZFENHADDR R/W Upper bound of Java2 R/W protection fence 0x040000c0 BUS_JZFENHADDR R/W Cancels writes to the J1 and J2 valid address ranges 0x040000c0 BUS_STOPOFRAM R/W CPU reset string 0x040000c0 BUS_USEBOOTMODE R/W HW/SW system configuration 0x04001000	0x0400019c	BUS_TIMINTSTAT	Clr	Timing interrupt status - clear	
0x040000a8 RESETCAUSE R/Set Reset cause - set 0x040000ac RESETCAUSE Clr Reset cause - clear 0x040000b0 BUS_JIFENLADDR R/W Lower bound of Java1 R/W protection fence 0x040000b4 BUS_JIFENHADDR R/W Upper bound of Java2 R/W protection fence 0x040000b6 BUS_J2FENHADDR R/W Lower bound of Java2 R/W protection fence 0x040000c0 BUS_J2FENHADDR R/W Upper bound of Java2 R/W protection fence 0x040000c0 BUS_J2FENHADDR R/W Upper bound of Java2 R/W protection fence 0x040000c0 BUS_J2FENHADDR R/W Size of the physical RAM 0x040000c0 BUS_TOPOFRAM R/W Cancels writes to the J1 and J2 valid address ranges 0x040000c4 BUS_BOOTMODE R/W CPU reset string 0x040000c6 BUS_USEBOOTMODE R/W Enable BOOTMODE rioUnit Registers R/W HW/SW system configuration 0x04001000 RIO_SYSCONFIG R/W HW/SW system configuration 0x04001000 ROM_CNTL1 R/W ROM control for Bank 1	0x040000a4	BUS_TIMINTEN	R/Set	Timing interrupt enable - set	
0x040000ac RESETCAUSE CIr Reset cause - clear 0x040000b0 BUS_JIFENLADDR R/W Lower bound of Java1 R/W protection fence 0x040000b4 BUS_JIFENHADDR R/W Upper bound of Java2 R/W protection fence 0x040000b8 BUS_J2FENLADDR R/W Lower bound of Java2 R/W protection fence 0x040000c0 BUS_J2FENHADDR R/W Upper bound of Java2 R/W protection fence 0x040000c0 BUS_TOPOFRAM R/W Size of the physical RAM 0x040000c4 BUS_FENCECNTL R/W Cancels writes to the J1 and J2 valid address ranges 0x040000c8 BUS_BOOTMODE R/W CPU reset string 0x040000cc BUS_USEBOOTMODE R/W Enable BOOTMODE rioUnit Registers R/W HW/SW system configuration 0x04001000 RIO_SYSCONFIG R/W HW/SW system configuration 0x04001004 ROM_CNTL0 R/W ROM control for Bank 0 0x04001008 ROM_CNTL1 R/W ROM control for Device 0+3 0x04001000 DEV_CNTL1 R/W ISA control for Device 2+1 0x04001014 RESERVAD N/A N/A <td< td=""><td>0x040001a4</td><td>BUS_TIMINTEN</td><td>Clr</td><td>Timing interrupt enable - clear</td></td<>	0x040001a4	BUS_TIMINTEN	Clr	Timing interrupt enable - clear	
DXO40000b0 BUS_J1FENLADDR R/W Lower bound of Java1 R/W protection fence	0x040000a8	RESETCAUSE	R/Set	Reset cause - set	
fence	0x040000ac	RESETCAUSE	Clr	Reset cause - clear	
Fence	0x040000b0	BUS_J1FENLADDR	R/W		
Series S	0x040000b4	BUS_J1FENHADDR	R/W		
0x040000c0 BUS_TOPOFRAM R/W Size of the physical RAM 0x040000c4 BUS_FENCECNTL R/W Cancels writes to the J1 and J2 valid address ranges 0x040000c8 BUS_BOOTMODE R/W CPU reset string 0x040000cc BUS_USEBOOTMODE R/W Enable BOOTMODE rioUnit Registers 0x04001000 RIO_SYSCONFIG R/W HW/SW system configuration 0x04001004 ROM_CNTL0 R/W ROM control for Bank 0 0x04001008 ROM_CNTL1 R/W ROM control for Bank 1 0x0400100c DEV_CNTL0 R/W ISA control for Device 0+3 0x04001010 DEV_CNTL1 R/W ISA control for Device 2+1 0x04001014 RESERVED N/A 0x04001018 BUS_WDREG_C W Watchdog count reset 0x04001010 RIO_CNTL R/W RIO bus turnaround control audUnit Registers 0x04002000 AUD_OCSTART R/W Starting byte address of current DMA output transfer	0x040000b8	BUS_J2FENLADDR	R/W		
0x040000c4 BUS_FENCECNTL R/W Cancels writes to the J1 and J2 valid address ranges 0x040000c8 BUS_BOOTMODE R/W CPU reset string 0x040000cc BUS_USEBOOTMODE R/W Enable BOOTMODE rioUnit Registers Enable BOOTMODE R/W Enable BOOTMODE 0x04001000 RIO_SYSCONFIG R/W HW/SW system configuration 0x04001004 ROM_CNTL0 R/W ROM control for Bank 0 0x04001008 ROM_CNTL1 R/W ROM control for Device 0+3 0x0400100c DEV_CNTL0 R/W ISA control for Device 2+1 0x04001014 RESERVHD N/A ISA control for WebTV Port 0x04001018 BUS_WDREG_C W Watchdog count reset 0x04001020 RIO_CNTL R/W RIO bus turnaround control audUnit Registers 0x04002000 AUD_OCSTART R/W Starting byte address of current DMA output transfer	0x040000bc	BUS_J2FENHADDR	R/W		
0x040000c8BUS_BOOTMODER/WCPU reset string0x040000ccBUS_USEBOOTMODER/WEnable BOOTMODErioUnit Registers0x04001000RIO_SYSCONFIGR/WHW/SW system configuration0x04001004ROM_CNTL0R/WROM control for Bank 00x04001008ROM_CNTL1R/WROM control for Bank 10x0400100cDEV_CNTL0R/WISA control for Device 0+30x04001010DEV_CNTL1R/WISA control for Device 2+10x04001014RESHRVHDIN/AIN/A0x04001016BUS_WDREG_CWWatchdog count reset0x04001010WTV_CNTLR/WControl for WebTV Port0x04001020RIO_CNTLR/WRIO bus turnaround controlaudUnit Registers0x04002000AUD_OCSTARTR/WStarting byte address of current DMA output transfer	0x040000c0	BUS_TOPOFRAM	R/W	Size of the physical RAM	
0x040000cc BUS_USEBOOTMODE R/W Enable BOOTMODE rioUnit Registers 0x04001000 RIO_SYSCONFIG R/W HW/SW system configuration 0x04001004 ROM_CNTL0 R/W ROM control for Bank 0 0x04001008 ROM_CNTL1 R/W ROM control for Bank 1 0x0400100c DEV_CNTL0 R/W ISA control for Device 0+3 0x04001010 DEV_CNTL1 R/W ISA control for Device 2+1 0x04001014 RESERVED N/A 0x04001018 BUS_WDREG_C W Watchdog count reset 0x0400101c WTV_CNTL R/W Control for WebTV Port 0x04001020 RIO_CNTL R/W RIO bus turnaround control audUnit Registers 0x04002000 AUD_OCSTART R/W Starting byte address of current DMA output transfer	0x040000c4	BUS_FENCECNTL	R/W	Cancels writes to the J1 and J2 valid address ranges	
rioUnit Registers 0x04001000 RIO_SYSCONFIG R/W HW/SW system configuration 0x04001004 ROM_CNTL0 R/W ROM control for Bank 0 0x04001008 ROM_CNTL1 R/W ROM control for Bank 1 0x0400100c DEV_CNTL0 R/W ISA control for Device 0+3 0x04001010 DEV_CNTL1 R/W ISA control for Device 2+1 0x04001014 RESHRVRD N/A 0x04001018 BUS_WDREG_C W Watchdog count reset 0x04001010 WTV_CNTL R/W Control for WebTV Port 0x04001020 RIO_CNTL R/W RIO bus turnaround control audUnit Registers 0x04002000 AUD_OCSTART R/W Starting byte address of current DMA output transfer	0x040000c8	BUS_BOOTMODE	R/W	CPU reset string	
0x04001000RIO_SYSCONFIGR/WHW/SW system configuration0x04001004ROM_CNTL0R/WROM control for Bank 00x04001008ROM_CNTL1R/WROM control for Bank 10x0400100cDEV_CNTL0R/WISA control for Device 0+30x04001010DEV_CNTL1R/WISA control for Device 2+10x04001014RESHRVHDN/A0x04001016BUS_WDREG_CWWatchdog count reset0x0400101cWTV_CNTLR/WControl for WebTV Port0x04001020RIO_CNTLR/WRIO bus turnaround controlaudUnit Registers0x04002000AUD_OCSTARTR/WStarting byte address of current DMA output transfer	0x040000cc	BUS_USEBOOTMODE	R/W	Enable BOOTMODE	
0x04001004 ROM_CNTL0 R/W ROM control for Bank 0 0x04001008 ROM_CNTL1 R/W ROM control for Bank 1 0x0400100c DEV_CNTL0 R/W ISA control for Device 0+3 0x04001010 DEV_CNTL1 R/W ISA control for Device 2+1 0x04001014 RESHRVHD N/A N/A 0x04001018 BUS_WDREG_C W Watchdog count reset 0x0400101c WTV_CNTL R/W Control for WebTV Port 0x04001020 RIO_CNTL R/W RIO bus turnaround control audUnit Registers R/W Starting byte address of current DMA output transfer	rioUnit Registe	ers			
0x04001008 ROM_CNTL1 R/W ROM control for Bank 1 0x0400100c DEV_CNTL0 R/W ISA control for Device 0+3 0x04001010 DEV_CNTL1 R/W ISA control for Device 2+1 0x04001014 RESHRVRD N/A 0x04001018 BUS_WDREG_C W Watchdog count reset 0x0400101c WTV_CNTL R/W Control for WebTV Port 0x04001020 RIO_CNTL R/W RIO bus turnaround control audUnit Registers 0x04002000 AUD_OCSTART R/W Starting byte address of current DMA output transfer	0x04001000	RIO_SYSCONFIG	R/W	HW/SW system configuration	
0x0400100c DEV_CNTL0 R/W ISA control for Device 0+3 0x04001010 DEV_CNTL1 R/W ISA control for Device 2+1 0x04001014 RESHRVRID. N/A 0x04001018 BUS_WDREG_C W Watchdog count reset 0x0400101c WTV_CNTL R/W Control for WebTV Port 0x04001020 RIO_CNTL R/W RIO bus turnaround control audUnit Registers 0x04002000 AUD_OCSTART R/W Starting byte address of current DMA output transfer	0x04001004	ROM_CNTL0	R/W	ROM control for Bank 0	
0x04001010 DEV_CNTL1 R/W ISA control for Device 2+1 0x04001014 RESHRVRD N/A 0x04001018 BUS_WDREG_C W Watchdog count reset 0x0400101c WTV_CNTL R/W Control for WebTV Port 0x04001020 RIO_CNTL R/W RIO bus turnaround control audUnit Registers 0x04002000 AUD_OCSTART R/W Starting byte address of current DMA output transfer	0x04001008	ROM_CNTL1	R/W	ROM control for Bank 1	
0x04001014 RESHRVRD N/A 0x04001018 BUS_WDREG_C W Watchdog count reset 0x0400101c WTV_CNTL R/W Control for WebTV Port 0x04001020 RIO_CNTL R/W RIO bus turnaround control audUnit Registers 0x04002000 AUD_OCSTART R/W Starting byte address of current DMA output transfer	0x0400100c	DEV_CNTL0	R/W	ISA control for Device 0+3	
0x04001018 BUS_WDREG_C W Watchdog count reset 0x0400101c WTV_CNTL R/W Control for WebTV Port 0x04001020 RIO_CNTL R/W RIO bus turnaround control audUnit Registers audUnit Registers R/W Starting byte address of current DMA output transfer	0x04001010	DEV_CNTL1	R/W	2 20 24 9 9 40 107 2 42 107 2	
0x0400101c WTV_CNTL R/W Control for WebTV Port 0x04001020 RIO_CNTL R/W RIO bus turnaround control audUnit Registers 0x04002000 AUD_OCSTART R/W Starting byte address of current DMA output transfer	0x04001014	RESERVIED.	N/A		
0x04001020 RIO_CNTL R/W RIO bus turnaround control audUnit Registers 0x04002000 AUD_OCSTART R/W Starting byte address of current DMA output transfer	0x04001018	BUS_WDREG_C	W	Watchdog count reset	
audUnit Registers 0x04002000 AUD_OCSTART R/W Starting byte address of current DMA output transfer	0x0400101c	WTV_CNTL	R/W	Control for WebTV Port	
0x04002000 AUD_OCSTART R/W Starting byte address of current DMA output transfer	0x04001020	RIO_CNTL	R/W	RIO bus turnaround control	
output transfer	audUnit Registers				
0x04002004 AUD_OCSIZE R/W Size of current DMA output transfer	0x04002000	AUD_OCSTART	R/W	Starting byte address of current DMA output transfer	
	0×04002004	AUD_OCSIZE	R/W	Size of current DMA output transfer	

 Table 5-1
 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x04002008	AUD_OCCONFIG	R/W	Configuration of current audio output DMA channel
0x0400200c	AUD_OCCNT	R/W	Byte count of current DMA output transfer
0x04002010	AUD_ONSTART	R/W	Starting byte address of next DMA output transfer
0x04002014	AUD_ONSIZE	R/W	Size of next DMA output transfer
0x04002018	AUD_ONCONFIG	R/W	Audio output DMA channel configuration
0x0400201c	AUD_ODMACNTL	R/W	Audio output DMA channel control
0x04002020	AUD_ICSTART	R/W	Starting byte address of current DMA input transfer
0x04002024	AUD_ICSIZE	R/W	Size of current DMA input transfer
0x04002028	RHSHRVHD	N/A	新疆,这一个一个一个一个一个一个一个一个一个一个一个一个一个一个一个一个一个一个一个
0x0400202c	AUD_ICCNT	R/W	Byte count of current DMA input transfer
0x04002030	AUD_INSTART	R/W	Starting byte address of next DMA input transfer
0x04002034	AUD_INSIZE	R/W	Size of next DMA input transfer
0x04002038=	RESERVED	N/A	
0x0400203c	AUD_IDMACNTL	R/W	Audio input DMA channel control
0×04002040	AUD_FCNTL	R/W	Codec clock and data format control
0x04002044	AUD_GPOCNTL	R/W	GPIO output control
0x021020282 0x02102158	RESIBEVIED	N/A	
0x0400205c	AUD_IODMACNTL	R/W	Shadow of audio I/O DMA channel control
vidunit Registe	ers		
0x04003000	VID_CSTART	R/W	Starting word address of current video DMA transfer
0x04003004	VID_CSIZE	R/W	Size of current video DMA transfer
0x04003008	VID_CCNT	R/W	Byte count of current video DMA transfer
0x0400300c	VID_NSTART	R/W	Starting word address of next video DMA transfer

 Table 5-1
 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x04003010	VID_NSIZE	R/W	Size of next video DMA transfer
0x04003014	VID_DMACNTL	R/W	Video DMA channel control
0x04003044- 0x04003134	RESERVED	N/A	
0x04003038	VID_INTSTAT	R/Set	vidUnit interrupt status - set
0x04003138	VID_INTSTAT	Clr	vidUnit interrupt status - clear
0x0400303c	VID_INTEN	R/Set	vidUnit interrupt enable - set
0x0400313c	VID_INTEN	Clr	vidUnit interrupt enable - clear
0x04003040	VID_VDATA	R/Wr	Video data
devUnit Regist	ers		
0x04004000	DEV_IROLD	R/W	Input data from external IR controller
0x04004004	DEV_LED	R/W	LED control
0x04004008	DEV_IDCNTL	R/W	ID chip control
0x0400400c	DEV_IICCNTL	R/W	IIC bus control
0x04004010	DEV_GPIOIN	R/W	GPIO input value
0x04004014	DEV_GPIOOUT	R/Set	GPIO output value - set
0x04004114	DEV_GPIOOUT	Clr	GPIO output value - clear
0x04004018	DEV_GPIOEN	R/Set	GPIO I/O configuration - set
0x04004118	DEV_GPIOEN	Clr	GPIO I/O configuration - clear
0x04004020	DEV_IRIN_SAMPLE_INT	R/W	IR receiver sample clock timing
0x04004024	DEV_IRIN_REJECT_INT	R/W	IR signal rejection timing
0x04004028	DEV_IRIN_TRANS_ DATA	R/W	IR transition data
0x0400402c	DEV_IRIN_STATCNTL	R/W	IR receiver status
0x04004030=== 0x0400403===	(RASERVIED)	N/A	
0x04004040	DEV_IROUT_FIFO	R/W	IR output data
0x04004044	DEV_IROUT_STATUS	R/W	IR output status
0x04004048	DEV_IROUT_PERIOD	R/W	Subcarrier clock period
0x0400404c	DEV_IROUT_ON	R/W	Time (in clocks) that IR LED is on each period
0x04004050	DEV_IROUT_CURRENT_ PERIOD	R/W	Time left in current subcarrier period

 Table 5-1
 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x04004054	DEV_IROUT_CURRENT_ ON	R/W	Time left for IR LED to be on
0x04004058	DEV_IROUT_CURRENT_ COUNT	R/W	Number of subcarrier periods left for the current operation
0x0400405c- 0x040041ff	RESERVED	N/A	
0x04004200	DEV_PPORT_DATA	R/W	Parallel Port data
0x04004204	DEV_PPORT_CTRL	R/W	Parallel Port control
0x04004208	DEV_PPORT_STAT	R/W	Parallel Port status
0x0400420c	DEV_PPORT_CNFG	R/W	Parallel Port configuration
0x04004210	DEV_PPORT_FIFOCTRL	R/W	Parallel Port FIFO interrupt and pointer control
0x04004214	DEV_PPORT_FIFOSTAT	R/W	Parallel Port FIFO status
0x04004218	DEV_PPORT_TIMEOUT	R/W	Parallel Port timeout threshold
0x0400421c	DEV_PPORT_STAT2	R/W	Parallel Port handshake signal levels
0x04004220	DEV_PPORT_IEN	R/W	Parallel Port interrupt enable
0x04004224	DEV_PPORT_IST	R/W	Parallel Port interrupt status
0x04004228	DEV_PPORT_CLRINT	R/W	Clear Parallel Port interrupt
0x0400422c	DEV_PPORT_ENABLE	R/W	Parallel Port enable/disable
0x04004800	RESERVED	N/A	
0x04004804	DEV_DIAG	R/W	Diag bus control
0x04004808	DEV_DEVDIAG	R/W	Debug data select for devUnit
0804004HE	RESERVED AND COMPANY		
memUnit Regi	sters		
0x04005000	MEM_TIMING	R/W	SDRAM timing parameters
0x04005004	MEM_CNTL	R/W	memUnit control
0x04005008	MEM_BURP	R/W	Memory access arbitration control
0x0400500c	MEM_REFCNTL	R/W	SDRAM refresh control
0×04005010	MEM_CMD	R/W	SDRAM commands
0x0400501c= 0 0x04006000	RESERVED	N/A	

 Table 5-1
 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description			
gfxUnit Regist	gfxUnit Registers					
0x04006004	GFX_CONTROL	Rd, Wr	gfxUnit configuration			
0x04006008- 0x0400600c	RESERVED	N/A				
0x04006010	GFX_OOTYCOUNT	R/W	yCounter line count value			
0x04006014	GFX_CELSBASE	R/W	Pointer to first CelRecord			
0x04006018	GFX_YMAPBASE	R/W	Pointer to yMap structure			
0x0400601c	GFX_CELSBASEMASTER	R/W	CelBaseMaster value			
0x04006020	GFX_YMAPBASEMASTER	R/W	yMapBaseMaster value			
0x04006024	GFX_INITCOLOR	R/W	Default buffer color value			
0x04006028	GFX_YCOUNTERINIT	R/W	Line counter's init value at VSYNC			
0x0400602c	GFX_PAUSECYCLES	R/W	Number of inactive cycles in each line			
0x04006030	GFX_OOTCELSBASE	R/W	CelBase value at last OOT interrupt			
0x04006034	GFX_OOTYMAPBASE	R/W	yMapBase value at last OOT interrupt			
0x04006038	GFX_OOTCELSOFFSET	R/W	Value of the CelsOffset register at the time of the last OOT interrupt			
0x0400603c	GFX_OOTYMAPCOUNT	R/W	yMapCount value at last OOT interrupt			
0x04006040	GFX_TERMCYCLECOUNT	R/W	Time required to compose last horizontal line			
0x04006044	GFX_HCOUNTERINIT	R/W	One half the number of pixels on an active line			
0x04006048	GFX_BLANKLINES	R/W	Number of vertical blanking lines that gfxUnit is off the bus			
0x0400604c	GFX_ACTIVELINES	R/W	Number of active video lines in a field/ frame			
0x04006050= 0x0400605c=	RESERVED	N/A				
0x04006060	GFX_INTEN	R/W	gfxUnit interrupt enable			
0x04006064	GFX_INTEN	Clr	gfxUnit interrupt enable - clear			
0x04006068	GFX_INTSTAT	R/W	gfxUnit interrupt status			
0x0400606c	GFX_INTSTAT	Clr	gfxUnit interrupt status - clear			
0x04006080	GFX_WBDSTART	R/W	Starting address of write-back buffer			
0x04006084	GFX_WBDLSIZE	R/W	Width of the write-back frame buffer (i.e., number of pixels in one scan line)			

Table 5-1 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x04006088	RESERVED	N/A	
0x0400608c	GFX_WBSTRIDE	R/W	Offset between frame buffer lines
0x04006090	GFX_WBDCONFIG	R/W	Write-back configuration bits
0x04006094	GFX_WBDSTART	R/W	Specifies where in memory the frame buffer data will be written by the writeback unit
dveUnit Regis	ters	-	
0x04007000	DVE_CNTL	R/W	dveUnit control
0x04007004	DVE_CNFG	R/W	dveUnit configuration
0x04007008	DVE_DBDATA	R/W	Debug and synchronization values for dveUnit counters
0x0400700c	DVE_DBEN	R/W	Enables loading of DBDATA register
0x04007010	DVE_DTST	R/W	DAC control
0x04007014	DVE_RDFIELD	R	Current dveUnit counter values
0x04007018	DVE_RDPHASE	R	Current color burst phase value
0x0400701c	DVE_FILTCNTL	R/W	Output filter control
0x04007/020 0x64007/ff	RESERVED		
divUnit Regis		1500	T-mara in the same in
0x04008000	DIV_SYNCCNTL	R/W	HSYNC and VSYNC control
0x04008004	DIV_DMACNTL	R/W	DMA control
0x04008008	DIV_NEXTVBITB	R/W	Starting and ending line numbers for VBI DMA
0x0400800c	DIV_NEXTVBILR	R/W	Number of clocks, following HSYNC, before next first valid pixel appears at data output during next DMA opera- tion
0x04008010	DIV_NEXTVBIADDR	R/W	Starting memory address where VBI data will be written
0×04008014	DIV_NEXTTB	R/W	Starting and ending active video lines of next video DMA
0x04008018	DIV_NEXTLR	R/W	Number of clocks, following HSYNC, before next first valid pixel appears at data input during next DMA operation

 Table 5-1
 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x0400801c	DIV_NEXTCFG	R/W	Which field is captured by the next DMA operation
0x04008020	DIV_NEXTADDR	R/W	Starting address of next DMA operation
0x04008024	DIV_CURRVBITB	R/W	Starting and ending line numbers for VBI DMA operation
0x04008028	DIV_CURRVBILR	R/W	Number of clocks, following HSYNC, before next first valid pixel appears at data output during current DMA oper- ation
0x0400802c	DIV_CURRVBIADDR	R/W	Starting address where VBI data will be written
0x04008030	DIV_CURRTB	R/W	Starting and ending active video lines for current video DMA
0x04008034	DIV_CURRLR	R/W	Number of clocks, following HSYNC, before next first valid pixel appears at data input during current DMA opera- tion
0x04008038	DIV_CURRCFG	R/W	Which field is captured by the current DMA operation
0x0400803c	DIV_CURRADDR	R/W	Starting address for current DMA operation
0x04008040	DIV_AUDCNTL	R/W	Audio signal timing
0x04008044	DIV_NEXTAUDADDR	R/W	Next audio data address
0x04008048	DIV_NEXTAUDLEN	R/W	Length of next audio sample
0x0400804c	DIV_CURRAUDADDR	R/W	Current audio data address
0x04008050	DIV_CURRAUDLEN	R/W	Length of current audio sample
0x04008054 0x0400805c	RESURVIED	N/A	
0x04008060	DIV_SYNCPHASE	R/W	Input and output video sync
0x04008064	DIV_GPOEN	R/W	GPIO reset
0x04008068	DIV_GPO	R/W	GPIO output values
0x0400806c	DIV_GPI	R/W	GPIO input values
0x04008070 0x0490907c	RESPRICED	N/A	
potUnit Registe			
0x04009080	POT_VSTART	R/W	Vertical starting line of active screen

 Table 5-1
 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x04009084	POT_VSIZE	R/W	Vertical size of the active screen
0x04009088	POT_BLNKCOL	R/W	Default color displayed during blank times
0x0400908c	POT_HSTART	R/W	Horizontal starting pixel of active screen
0x04009090	POT_HSIZE	R/W	Horizontal size of the active screen
0x04009094	POT_CNTL	R/W	potUnit control
0x04009098	POT_HINTLINE	R/W	Line that caused an horizontal interrupt
0x0400909c	POT_INTEN	R/Set	potUnit interrupt enable - set
0x040090a4	POT_INTEN	Clr	potUnit interrupt enable - clear
0x040090a0	POT_INTSTAT	R/Set	potUnit interrupt status - set
0x040090a8	POT_INTSTAT	Clr	potUnit interrupt status - clear
0x040090ac	POT_CLINE	R/W	Current line being diaplayed
(0x040090505) 0x04009ffc	RESERVED	N/A	
sucUnit Regis	ters	,	
0x0400a000	SUCGPU_TFFHR	R/W	Transmit FIFO data
0x0400a004	SUCGPU_TFFHRSRW	R/W	Same as TFFHR register, except used for debugging
0x0400a008	SUCGPU_TFFTRG	R/W	Transmit FIFO trigger level
0x0400a00c	SUCGPU_TFFCNT	R/W	Current number of entries in Transmit FIFO
0x0400a010	SUCGPU_TFFMAX	R/W	Maximum Transmit FIFO depth
0x0400a014	SUCGPU_TFFCR	R/W	Transmit FIFO control
0x0400a018	SUCGPU_TFFSR	R/W	Transmit FIFO status
0x0400a01c	SUCGPU_TFFGCR	R/W	Transmit and receive FIFO size is swapped
0x0400a020=	RIESUR(VED)	N/A	
0x0400ä03c			
0x0400a040	SUCGPU_RFFHR	R/W	Receive FIFO data
0x0400a044	SUCGPU_RFFHRSRW	R/W	Same as RFFHR register, except used for debugging
0x0400a048	SUCGPU_RFFTRG	R/W	Receive FIFO trigger level

Table 5-1 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x0400a04c	SUCGPU_RFFCNT	R/W	Current number of entries in Receive FIFO
0x0400a050	SUCGPU_RFFMAX	R/W	Maximum Receive FIFO depth
0x0400a054	SUCGPU_RFFCR	R/W	Receive FIFO control
0x0400a058	SUCGPU_RFFSR	R/W	Receive FIFO status
0x0400a05c	SUCGPU_RFFGCR	R/W	Transmit and receive FIFO size is swapped
0x0400a060 0x0400a07c	RESERVED	N/A	
0x0400a080	SUCGPU_TSRCR	R/W	Transmit shift register control
0x0400a084	SUCGPU_TSRSTATE	R/W	Transmit shift register state machine
0x0400a088	SUCGPU_TSRBCCNT	R	Current state of the transmit bit clock counter
0x0400a08c	SUCGPU_TSRBITCNT	R	Current state of the transmit data bit counter
0x0400a090-1	RESERVED	N/A a	
0x0400a0bc			
0x0400a0c0	SUCGPU_RSRCR	R/W	Receive shift register control
0x0400a0c4	SUCGPU_RSRSTATE	R/W	Receive shift register state machine
0x0400a0c8	SUCGPU_RSRBCCNT	R	Current state of the receive bit clock counter
0x0400a0cc	SUCGPU_RSRBITCNT	R	Current state of the receive bit data counter
0x0400a0d0= 0x0400a0fc	RESHRVED.	N/A	
0x0400a100	SUCGPU_MCD	R/W	Master clock divisor
0x0400a104	SUCGPU_SCD0	R/W	Sample clock divisor
0x0400a108	SUCGPU_SCD1	R/W	Sample clock divisor
0x0400a10c	SUCGPU_CCR	R/W	Smart Card master and sample clock control
0x0400a17c	RESERVED	N/A	
0x0400a180	SUCGPU_LCR	R/W	Transmit and receive data line control

 Table 5-1
 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x0400a184	SUCGPU_LSCR	R/W	Transmit and receive data line control specific to Smart Cards
0x0400a188	SUCGPU_LSTPBITS	R/W	Number of Stop bits
0x0400a18c	SUCGPU_LSR	R/W	Transmit and receive data transfer status
0x0400a190- 0x0400a1fc	RESERVED	N/A	
0x0400a200	SUCGPU_ISR	W	Interrupt status
0x0400a204	SUCGPU_ISR	R	Interrupt status - reset
0x0400a208	SUCGPU_IER	Set	Interrupt enable - set
0x0400a20c	SUCGPU_IER	Clr	Interrupt enable - clear
0x0400a210	SUCGPU_IER	R/W	Interrupt enable
0x0400a214 0x0400a2/c	RESERVED	N/A	
0x0400a280	SUCGPU_SDSMCR	R/W	Deactivate Smart Card
0x0400a284	SUCGPU_SDSMSTATE	R/W	Smart Card state machine
0x0400a288	SUCGPU_IOOD	R/W	GPIO open-drain control
0x0400a28c	SUCGPU_SPIOCR	R/W	GPIO deactivate state machine
0x0400a290	SUCGPU_SPIOEN	Set	Special-purpose enables - set
0x0400a294	SUCGPU_SPIOEN	Clr	Special-purpose enables - clear
0x0400a298	SUCGPU_SPIOEN	R/W	Special-purpose enables
0x0400a29ca	RESERVED	N/A	
0x0400a2a0	SUCGPU_GPIODIR	Set	GPIO line direction control - set
0x0400a2a4	SUCGPU_GPIODIR	Clr	GPIO line direction control - clear
0x0400a2a8	SUCGPU_GPIODIR	R/W	GPIO line direction control
0x0400a2ac	Residence of the second	N/A	
0x0400a2b0	SUCGPU_GPIOVAL	Set	GPIO line values - set
0x0400a2b4	SUCGPU_GPIOVAL	Clr	GPIO line values - clear
0x0400a2b8	SUCGPU_GPIOVAL	R/W	GPIO line values
0x0400a2bc	RESERVED:	NVA"-	
0x0400a2c0	SUCGPU_GPIORIER	Set	GPIO interrupt enable - set
0x0400a2c4	SUCGPU_GPIORIER	Clr	GPIO interrupt enable - clear
0x0400a2c8	SUCGPU_GPIORIER	R/W	GPIO interrupt enable

 Table 5-1
 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x0400a2cc	RESERVED	N/A	
0x0400a2d0	SUCGPU_GPIOFIER	Set	GPIO interrupt enable - set
0x0400a2d4	SUCGPU_GPIOFIER	Clr	GPIO interrupt enable - clear
0x0400a2d8	SUCGPU_GPIOFIER	R/W	GPIO interrupt enable
0x0400_a2dc	RESERVED	N/A	
0x0400a2e0	SUCGPU_GPIOISR	Set	UART interrupt status - set
0x0400a2e4	SUCGPU_GPIOISR	Clr	UART interrupt status - clear
0x0400a2e8	SUCGPU_GPIOISR	R/W	UART interrupt status
0x0400a2ec	SUCGPU_GPIORISR	R/W	UART interrupt status
0x0400a2f0	SUCGPU_GPIOFISR	R/W	UART interrupt status
0x0400a2f4 0x0400a2fc	RESERVED	N/A	
0x0400a300	SUCGPU_DIAGMODE	R/W	Diagnostic mode control
0x0400a304 0x0400a7fc	RESERVED	N/A	
0x0400a800	SUCSC0_TFFHR	R/W	Smart Card Receive FIFO data
0x0400a804	SUCSC0_TFFHRSRW	R/W	Same as TFFHR register, except used for debugging
0x0400a808	SUCSC0_TFFTRG	R/W	Smart Card Transmit FIFO trigger level
0x0400a80c	SUCSC0_TFFCNT	R/W	Current number of entries in Smart Card Receive FIFO
0x0400a810	SUCSC0_TFFMAX	R/W	Maximum Smart Card Receive FIFO depth
0x0400a814	SUCSC0_TFFCR	R/W	Smart Card Transmit FIFO control
0x0400a818	SUCSC0_TFFSR	R/W	Smart Card Transmit FIFO status
0x0400a81c	SUCSC0_TFFGCR	R/W	Smart Card Transmit and Receive FIFO size is swapped
0x0400a820- 0x0400a83c	RESERVED	N/A	
0x0400a840	SUCSC0_RFFHR	R/W	Smart Card Receive FIFO data
0x0400a844	SUCSC0_RFFHRSRW	R/W	Same as RFFHR register, except used for debugging
0x0400a848	SUCSC0_RFFTRG	R/W	Smart Card Receive FIFO trigger level

 Table 5-1
 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x0400a84c	SUCSC0_RFFCNT	R/W	Current number of entries in Smart Card Receive FIFO
0x0400a850	SUCSC0_RFFMAX	R/W	Maximum Smart Card Receive FIFO depth
0x0400a854	SUCSC0_RFFCR	R/W	Smart Card Receive FIFO control
0x0400a858	SUCSC0_RFFSR	R/W	Smart Card Receive FIFO status
0x0400a85c	SUCSC0_RFFGCR	R/W	Smart Card Transmit and Receive FIFO size is swapped
0x0400a860 0x0400a87c	RESURVED:	N/A	
0x0400a880	SUCSC0_TSRCR	R/W	Smart Card Transmit shift register control
0x0400a884	SUCSC0_TSRSTATE	R/W	Smart Card Transmit shift register state machine
0x0400a888	SUCSC0_TSRBCCNT	R	Current state of the transmit bit clock counter
0x0400a88c	SUCSC0_TSRBITCNT	R	Current state of the transmit bit data counter
0x0400a890 0x0400a8bc	RESERVED	N/A	
0x0400a8c0	SUCSC0_RSRCR	R/W	Smart Card Receive shift register control
0x0400a8c4	SUCSC0_RSRSTATE	R/W	Smart Card Receive shift register state machine
0x0400a8c8	SUCSC0_RSRBCCNT	R	Sample clock divisor
0x0400a8cc	SUCSC0_RSRBITCNT	R	Smart Card master and sample clock control
0x0400a8d0 0x0400a8fc	RESHRVED)		
0x0400a900	SUCSC0_MCD	R/W	Smart card master clock divisor
0x0400a904	SUCSC0_SCD0	R/W	Smart Card sample clock divisor
0x0400a908	SUCSC0_SCD1	R/W	Sample clock divisor
0x0400a90c	SUCSC0_CCR	R/W	Smart Card master and sample clock control
0x0400a910 0x0400a97c	RESHRVED.	N/A	

Table 5-1 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x0400a980	SUCSC0_LCR	R/W	Smart Card transmit and receive data line control
0x0400a984	SUCSC0_LSCR	R/W	Smart Card transmit and receive data line control specific to Smart Cards
0x0400a988	SUCSCO_LSTPBITS	R/W	Number of Stop bits
0x0400a98c	SUCSC0_LSR	R/W	Smart Card transmit and receive data transfer status
0x0400a990- 0x0400a99c	RESERVED	N/A	
0x0400aa00	SUCSC0_ISR	W	Smart Card interrupt status
0x0400aa04	SUCSC0_ISR	R	Smart Card interrupt status - reset
0x0400aa08	SUCSC0_IER	Set	Smart Card interrupt enable - set
0x0400aa0c	SUCSC0_IER	Clr	Smart Card interrupt enable - clear
0x0400aa10	SUCSC0_IER	R/W	Smart Card interrupt enable
0x0400aa14- 0x0400aa7c	RESERVED)	N/A	
0x0400aa80	SUCSC0_SDSMCR	R/W	Deactivate Smart Card
0x0400aa84	SUCSC0_SDSMSTATE	R/W	Smart Card state machine
0x0400aa88	SUCSC0_IOOD	R/W	Smart Card open-drain control
0x0400aa8c	SUCSC0_SPIOCR	R/W	Smart Card deactivate state machine
0x0400aa90	SUCSC0_SPIOEN	Set	Smart Card special-purpose enables - set
0x0400aa94	SUCSC0_SPIOEN	Clr	Smart Card special-purpose enables - clear
0x0400aa98	SUCSC0_SPIOEN	R/W	Smart Card special-purpose enables
0x0400aa9c	RESERVED	N/A	
0x0400aaa0	SUCSC0_GPIODIR	Set	Smart Card line direction control - set
0x0400aaa4	SUCSC0_GPIODIR	Clr	Smart Card line direction control - clear
0x0400aaa8	SUCSC0_GPIODIR	R/W	Smart Card line direction control
0x0400aaac	RESERVED.	N/A	
0x0400aab0	SUCSCO_GPIOVAL	Set	Smart Card line values - set
0x0400aab4	SUCSC0_GPIOVAL	Clr	Smart Card line values - clear
0x0400aab8	SUCSC0_GPIOVAL	R/W	Smart Card line values

Table 5-1 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x0400aabc	RESERVED	N/A	^{我一个} 不是一定的一个多数是数的基本。
0x0400aac0	SUCSCO_GPIORIER	Set	Smart Card interrupt enable - set
0x0400aac4	SUCSC0_GPIORIER	Clr	Smart Card interrupt enable - clear
0x0400aac8	SUCSC0_GPIORIER	R/W	Smart Card interrupt enable
0x0400aacc	RESERVED	N/A	
0x0400aad0	SUCSC0_GPIOFIER	Set	Smart Card interrupt enable - set
0x0400aad4	SUCSC0_GPIOFIER	Clr	Smart Card interrupt enable - clear
0x0400aad8	SUCSC0_GPIOFIER	R/W	Smart Card interrupt enable
0x0400aadc	RESERVED	N/A	
0x0400aae0	SUCSC0_GPIOISR	Set	Smart Card interrupt status - set
0x0400aae4	SUCSC0_GPIOISR	Clr	Smart Card interrupt status - clear
0x0400aae8	SUCSC0_GPIOISR	R/W	Smart Card interrupt status
0x0400aaec	SUCSC0_GPIORISR	R/W	Smart Card interrupt status
0x0400aaf0	SUCSC0_GPIOFISR	R/W	Smart Card interrupt status
0x0400aafc	TRESERVAND	N/A	
0x0400ab00	SUCSC0_DIAGMODE	R/W	Smart Card diagnostic mode control
0x0400als04 0x0400affc	RESERVED	N/A	
modUnit Regis	sters	A SANCE	
0x0400b000	MOD_OCSTART	R/W	Starting byte address of current DMA output transfer
0x0400b004	MOD_OCSIZE	R/W	Size of current output DMA transfer
0x0400b008	MOD_OCCONFIG	R/W	Not used
0x0400b00c	MOD_OCCNT	R/W	Byte count of current output DMA transfer
0x0400b010	MOD_ONSTART	R/W	Starting address of next output DMA transfer
0x0400b014	MOD_ONSIZE	R/W	Size of next output DMA transfer
0x0400b018	MOD_ONCONFIG	R/W	Next modem output DMA channel control
0x0400b01c	MOD_ODMACNTL	R/W	Modem output DMA channel control
0х0400ь020	MOD_ICSTART	R/W	Starting byte address of current input DMA transfer

 Table 5-1
 SOLO1 Registers - Arranged by Address (Continued)

Address	Register Name	R/W	Description
0x0400b024	MOD_ICSIZE	R/W	Size of current input DMA transfer
0x0400b028	RESERVED	N/A	
0x0400b02c	MOD_ICCNT	R/W	Byte count of current input DMA transfer
0x0400b030	MOD_INSTART	R/W	Starting address of next input DMA transfer
0х0400b034	MOD_INSIZE	R/W	Size of next input DMA transfer
0x0400b038	RESERVED	N/A	
0х0400b03c	MOD_IDMACNTL	R/W	Modem input DMA channel control
0х0400b040	MOD_FCNTL	R/W	Codec clock and data format control
0х0400b044	MOD_GPOCNTL	R/W	GPIO output control
-0x0400b048	RESERVED	N/A	
0x0400b05c	MOD_IODMACNTL	R/W	Shadow of modem input and output DMA channel control registers

5.2.2 busUnit Software Description

The busUnit provides an interface for an IDT r4640 CPU or a QED R5230 CPU to the rest of the system, including RAM, ROM, external devices and internal SOLO1 memory space.

The busUnit contains a number of registers that provide information about, or general control of SOLO1.

5.2.2.1 Reset Behavior

All interrupts are disabled at reset and all interrupt status bits are cleared, although interrupt events will be caught regardless of the values of their enable bits. Such events can be observed by polling the status registers that are not masked by enable bits. Or, if the enable bit is subsequently set, a prior interrupt event will immediately cause a CPU interrupt.

All fences are in a disabled state at reset.

The watchdog timer is disabled at reset. It's software reset (clear) value is set to 0x00 at reset.

ROM address space aliases onto RAM address space at reset. This must be disabled with the CHPCNTL register.

SOLO1 provides information about the cause of the most recent system reset. See the section on reset events for more information.

5.2.2.2 Interrupt Handling

This section describes how SOLO1 manages interrupts. Each of the registers referred to in this section are described in detail in the following subsections.

5.2.2.2.1 Top-level Interrupt Status

Interrupts in SOLO1 are implemented through a number of levels that collapse the various sources into one interrupt signal that is then delivered to the CPU. Interrupt sources are divided into groups roughly reflecting their type: video, audio, device, RIO (ISA) device, timer and fence.

The interrupt signal delivered to the CPU is active low, although this has no bearing on the programmer's model. Interrupt status registers in SOLO1 report interrupt events and are active high regardless of the polarity of the actual interrupt signal.

Note: Most interrupt sources are active low, although some are programmable in SOLO1 to accommodate sources of either polarity. The status bits, however, remain active high.

The first level of interrupt logic is represented by two read-only registers (INTSTAT and INTSTATRAW), which report the interrupt status of the six interrupt-type categories.

The INTSTAT register contains the main interrupt status masked by enable bits. This register is used by the interrupt handler to determine the source of a CPU interrupt. The INTSTATRAW register can be used to poll interrupt events even when the interrupt is disabled and would not result in an actual CPU interrupt.

The INTSTAT register contains the main interrupt status *not* masked by enable bits. This register does not have it's own enable register, nor can it be set or cleared. It is merely provided as a method for quickly determining which type of interrupt is being serviced. It's limit of six status bits and the fact that bits 1:0 are reserved as zero make it efficient to use a jump table to dispatch the next appropriate service routine.

5.2.2.2 Second-level Interrupt Status

Each of the main interrupt types has it's own set of interrupt registers. These consist of status and enable registers, and in some cases, a polarity register to facilitate use of both positive- and negative-polarity interrupt signals (see Table 5-2).

Register Name	Description	
xxxINTSTAT (R)	Interrupt status masked by enable bits	
xxxINTSTAT_S (RS)	Interrupt status "set" register and status *not* masked by enable bits	
xxxINTSTAT_C (C)	Interrupt status "clear" register	
xxxINTEN_S (RS)	Interrupt enable "set" register and read register for enables	
xxxINTEN_C (C)	Interrupt enable "clear" register	
xxxINTPOL (RW)	Interrupt polarity register	

Table 5-2 Interrupt Status and Enable Registers

Like the main interrupt status register, the secondary XXXINTSTAT register is used to determine the source of a CPU interrupt. Second-level interrupts may also be polled when not enabled by reading the XXXINTSTAT_S register, which also serves as the "set" register for the interrupt status. The ability to manually set interrupt status bits is provided mainly as a hardware debugging tool. Interrupt status bits are cleared through the XXXINTSTAT_C register, which is "clear" only and provides no read facility.

All second-level SOLO1 interrupts are enabled at the second level. The xxxINTEN_S and xxxINTEN_C registers provide "set" and "clear" functionality for the interrupt enable bits. xxxINTEN_S is also readable. In order for a

particular interrupt event to trigger a CPU interrupt, its enable bit must be set. However, even if an interrupt is not enabled, the interrupt event will be caught by the status register. This is visible through the xxxINTSTAT_S register, which provides interrupt status without regard to the enable bits. If an interrupt is enabled while it's status bit is set, the CPU will immediately see an interrupt.

Some of the second-level interrupts have a polarity register that allows software to set the polarity of the interrupt signal to match that of the corresponding device in the system. The xxxINTPOL register is read/write and has the same bit definitions as the other registers in it's group.

The second-level interrupt registers are listed in Table 5-2. Descriptions of each of these registers are provided later in this chapter.

Table 5-3 Second-level Interrupt Registers

Register Name Description			
Video Interrupts			
VIDINTSTAT (R)	0x0400_0080		
VIDINTSTAT_S (RS)	0x0400_0084		
VIDINTSTAT_C (C)	0x0400_0180		
VIDINTEN_S (RS)	0x0400_0088		
VIDINTEN_C (C)	0x0400_0188		
Audio Interrupts			
AUDINTSTAT (R)	0x0400_0068		
AUDINTSTAT_S (RS)	0x0400_006c		
AUDINTSTAT_C (C)	0x0400_0168		
AUDINTEN_S (RS)	0x0400_0070		
AUDINTEN_C (C)	0x0400_0170		
RIO (ISA) Interrupts			
RIOINTSTAT (R)	0x0400_008c		
RIOINTSTAT_S (RS)	0x0400_0090		
RIOINTSTAT_C (C)	0x0400_018c		
RIOINTEN_S (RS)	0x0400_0098		
RIOINTEN_C (C)	0x0400_0198		
RIOINTPOL (RW)	0x0400_0094		
SOLO1 Device Interrupts			
DEVINTSTAT (R)	0x0400_0074		

Table 5-3 Second-level Interrupt Registers (Continued)

Register Name	Description	
SOLO1 Device Interrupts (Continued)		
DEVINTSTAT_S (RS)	0x0400_0078	
DEVINTSTAT_C (C)	0x0400_0174	
DEVINTEN_S (RS)	0x0400_007c	
DEVINTEN_C (C)	0x0400_017c	
Timer Interrupts		
TIMINTSTAT (R)	0x0400_009c	
TIMINTSTAT_S (RS)	0x0400_00a0	
TIMINTSTAT_C (C)	0x0400_019c	
TIMINTEN_S (RS)	0x0400_00a4	
TIMINTEN_C (C)	0x0400_01a4	
Fence Interrupts		
FENINTSTAT (R)	0x0400_0010	
FENINTSTAT_S (RS)	0x0400_0054	
FENINTSTAT_C (C)	0x0400_0110	
FENINTEN_S (RS)	0x0400_0014	
FENINTEN_C (C)	0x0400_0114	

5.2.2.3 Third-level Interrupt Status

Some of the interrupt sources that occur at the second level of the interrupt hierarchy do not fully describe the interrupt event— just as the sources at the top level are not sufficient. The fence, timer and RIO interrupts are fully described by the second-level interrupt status registers. The video, audio and SOLO1 device interrupts contain at least one interrupt source which points to a third level of interrupt status.

The only third-level interrupt to reside in the busUnit is the GPIO interrupt. The GPIO interrupts have a set of registers similar to all of the second-level interrupts. Like second-level interrupts (and other third-level interrupts), the GPIO interrupts have their own enable registers and are "sticky." That is, in order for a GPIO to interrupt the CPU, it must be enabled both by its own third-level enable, as well as by the second-level enable in the SOLO1 device interrupt registers. When a GPIO interrupt is cleared, it must be cleared from the bottom up: first at the third-level GPIO interrupt status register, and then at the second-level SOLO1 device interrupt status register.

Note: The first interrupt level is not cleared since it is not a real status register, but simply a reflection of the second-level status registers.

Table 5-4 Third-level Interrupt Registers

Register Name	Description	
GPINTSTAT (R)	0x0400_0058	
GPINTSTAT_S (RS)	0x0400_0060	
GPINTSTAT_C (C)	0x0400_0158	
GPINTEN_S (RS)	0x0400_005c	
GPINTEN_C (C)	0x0400_015c	
GPINTPOL (RW)	0x0400_0064	

The first-level status register is not a register at all, but merely a reflection of the state of the second-level status registers. Hence, clearing a second-level status register results in the clearing of the first-level status register.

This ordering is necessary to prevent an uncleared interrupt at the third-level from triggering an interrupt at the second-level, even as the second-level interrupt is being cleared. The net result of clearing the third-level last would be a false interrupt which shows up at the second level, but not at the third.

It is also necessary to ensure that the CPU write transaction, which clears the second-level interrupt, is allowed to complete before the interrupt handler finishes. The CPU interrupt signal will go inactive on the cycle in which the write transaction takes place, but write buffering in both the CPU and SOLO1 may cause this to happen after the interrupt handler has returned control to the browser—causing a false interrupt to occur. The easiest way around this problem is to read the interrupt status register immediately after clearing it. The value returned is not important. The act of reading will have blocked the CPU from continuing until both the read (and the previous write) have finished.

A typical SOLO1-based system, which uses the IDT r4640 CPU, would have the interrupts connected to the CPU as shown in Table 5-5.

 Table 5-5
 Typical System Interrupt Configuration

Interrupt	Description
Interrupt 0	SOLO1 interrupt
Interrupt 1	Expansion Port (WebTV Port) interrupt
Interrupt 2	Not used
Interrupt 3	Not used

 Table 5-5
 Typical System Interrupt Configuration (Continued)

Interrupt	Description	
Interrupt 4	Diagnostic interrupt	
Interrupt 5	Reserved for CPU timer	
NMI	Imminent power loss interrupt	

The Non-Maskable Interrupt (NMI) is used to indicate that power will be lost no sooner than 2ms from when the interrupt is asserted. This time should be used to clean up important processes, such as disk accesses.

5.2.2.3.1 Fences

SOLO1 provides a number of memory fences which are used to flag memory accesses to special areas of the address map. At reset, the fences are collapsed to their smallest size and their interrupts are disabled.

Top-of-RAM Fence

The top-of-RAM fence is used to restrict RAM accesses to only that part of the address map which contains physical memory. At reset, this fence is set to 64MB, which is the most physical RAM that SOLO1 can support. Software is responsible for determining the exact amount of physical memory in the system (see the memUnit Registers section) and setting the top of the RAM fence if desired.

This fence is described by one control register (TOPOFRAM), which specifies the first invalid RAM address. RAM transactions to addresses below this mark are allowed to complete. RAM transactions at or above this mark generate bus errors. Writes are always cancelled. Note that killing writes can result in incoherent memory states between the CPU and physical RAM.

Unlike the other fences, the top-of-RAM fence does not have its own interrupts. It simply generates bus errors and it is up to software to check the address which generated the bus error and determine that it was above physical RAM. Note that, unlike the other fences which generate their own interrupts, bus errors are implemented as bus time-outs and as such take a long time to complete. The exact delay is settable by software, but will typically be ~750 microseconds. Hence, it is possible that an access above this fence will cause a delay long enough to disrupt some real-time process (video, audio and soft modem being the most likely.)

The top-of-RAM fence is not enabled or disabled except by setting it's address. The largest address in the SOLO1 RAM space is 0x03ff_ffff. Hence, setting this fence to an address above this maximum RAM address has the effect of turning it off, since there will never be RAM accesses above 0x03ff_ffff. The reset value of 0x0400_0000 means that this fence comes out of reset disabled.

Note that this fence is typically used to protect against accesses outside of the physical RAM space, that would otherwise wrap into physical RAM and cause mayhem. This fence can also be used to emulate systems with less memory (e.g., even if there is 8MB of physical RAM in the system, by setting this fence to 4MB, a 4MB system is constructed).

Low-memory Fences

The low-memory fences (one read and one write fence) restrict accesses to the lowest part of RAM. They are designed to operate correctly—even when RAM addresses wrap onto the lowest part of memory.

This tolerance of memory wrapping is superseded by using the top-of-RAM fence described earlier. Without the top-of-RAM fence, it was necessary to detect low memory accesses even if they resulted from overrunning the top of memory.

The bottom of the low-memory fences is implicitly 0x0000_0000 and the top is specified by control registers that allow up to 2K of RAM to be protected. The largest address which the fences can protect is 0x0000_07fc. Note that, unlike the top-of-RAM fence, the low-memory fences include the specified address (e.g., if the specified fence address is 0x0000_0040, then that address is protected by the low-memory fence, but 0x0000_0044 is not).

Masks are provided to specify which wrapping points are to be protected. Each bit of the mask represents a 1MB boundary in the address space. If the mask is set to $0x0000_0001$, then the memory space starting at $0x0000_0000$ is protected. If the mask is set to $0x0000_0003$, then the space starting at $0x0010_0000$ is also protected. Each mask bit is independent of the others. The fences are completely inactivated if the masks are set to $0x00000_0000$.

Both of the low-memory read and write fences have their own interrupts, that are used to indicate that the fence has been violated. Reads which violate the read fence are allowed to complete, but are flagged with an interrupt. Writes are cancelled, based on the value of the FENCECNTL register (see the FENCECNTL register description later in this section), but are also flagged with an interrupt.

Note: Cancelling writes can result in incoherent memory states between the CPU and physical RAM.

General-purpose Fences

The general-purpose fences are designed to protect arbitrary areas in RAM from reads, writes or both. Two fences are provided with different granularity. The first has cache-line granularity (8 long words). The second has 4K granularity.

Each fence is specified by a bottom address, which is included in the fence, and a top address, which is excluded from the fence. If the top and bottom of the fence are set to the same address, then the fence size is zero and no transactions will land within the fence. At reset, the top and bottom addresses of each fence are set to 0x0000_0000 and hence, neither fence is active.

Both general-purpose fences have their own read and write interrupts that are used to indicate that the fence has been violated. Reads that violate the fence are allowed to complete, but are flagged with an interrupt. Writes are cancelled based on the value of the FENCECNTL register (see the FENCECNTL register description later in this section) but are also flagged with an interrupt.

Note: Cancelling writes can result in incoherent memory states between the CPU and physical RAM.

The sense of each general-purpose fence can be reversed using the FENCECNTL register. Normally, transactions are flagged if they fall between the top and bottom addresses of the fence. If the sense of the fence is reversed, transactions between the top and bottom addresses are permitted and transactions beneath the bottom address or above (and including) the top address are flagged.

Fence Controls

The FENCECNTL register controls whether writes to any of the fences are allowed to continue (except for the top-of-RAM fence), and also controls the sense of the general-purpose fences.

Note: Writes above the top-of-RAM fence are always cancelled.

Fence checks are done on CPU transactions early in the pipe within SOLO1. This means that each fence has what amounts to a delay slot, which must be dealt with (i.e., if a write transaction is modifying one of the fence registers, then the subsequent transaction may be subject to the "old" fence instead of the modified fence). Whether or not this delay has any noticeable consequences, depends on the number of cycles between the transactions and the state of SOLO1's write buffer.

The safest way to avoid any problems related to these fence "delay slots" is to ensure that a write to one of the fence control registers is immediately followed by a read to the same register.

Note: Control register space cannot fall within any fences, so accesses to them do not depend on the actual state of the fence control registers.

Typical System Fence Configuration

A typical SOLO1-based system will contain 8MB of RAM. The top-of-RAM fence will be set accordingly:

TOPOFRAM<- 0x0080_0000

The low-memory fences will be used to protect low memory from de-referenced null pointers and from writes to the interrupt vector space. The low-memory fences will be set as follows:

LOWRDADDR<- 0x0000_00fc

LOWRDMASK<- 0x0000_0001

LOWWRADDR<- 0x0000_01fc

LOWWRMASK<- 0x0000_0001

The large granularity general-purpose fence (J2) will likely be used to protect code in RAM from stray writes. The small granularity general-purpose fence (J1) will likely be used by the Java VM.

Writes will be cancelled to low memory and to the fence protecting code space in RAM.

FENCECNTL<- 0x0000_0014

5.2.2.3.2 Timers

SOLO1 provides a number of timers, that operate in quite dissimilar ways and are used for dissimilar purposes.

System Timer

The system timer emulates the system timer of the IDT r4640 CPU (see page 5-4 of the IDT79R4640 Hardware User's Manual).

This timer consists of a count register and a compare register. The count register is read/write and increments every SOLO1 system clock cycle. The compare register contains a stable value. When the value of the compare register matches that of the count register, the system timer interrupt fires (refer to the "Interrupt Handling" section).

Writing the compare register (TCOUNT (RW) 0x0400_0048, TCOMPARE (RW) 0x0400_004c) has the side effect of clearing the system timer interrupt. Both registers are 32-bits wide. The reset value of each register is 0x0000_0000, although after reset, the TCOUNT register will begin incrementing each system clock cycle.

Watchdog Timer

The watchdog counter provides a mechanism to reset the system when software has crashed or otherwise lost control. This is implemented by a counter which causes a system reset if it expires. Software is responsible for resetting the counter at regular intervals to prevent such a reset from occurring, but if the software should crash, or otherwise be prevented from resetting the counter, the system will reset and return to a known state.

At reset, the watchdog is disabled. Enabling the watchdog requires a series of writes to the CHPCNTL register which must occur in the correct order. The WDENABLE field of the CHPCNTL register must be written the values 0x0, 0x1, 0x2 and 0x3 in that order to enable the watchdog. Disabling the watchdog requires the four values to be written in the reverse order: 0x3, 0x2, 0x1, 0x0. Transactions to other addresses, or to other control registers, are permitted between the transactions of the lock and unlock sequences.

Note that there are other fields in the CHPCNTL register which must not be disturbed while enabling or disabling the watchdog timer. Based on expected values of the various fields in the CHPCNTL register, the following can be used to enable the watchdog:

CHPCNTL<- 0x0000_ffff

CHPCNTL <- 0x4000_ffff

CHPCNTL<- 0x8000_ffff

CHPCNTL<- 0xc000_ffff

The watchdog timer also provides a mechanism to determine the length of a watchdog time-out. The watchdog counter counts up from 0x00 to 0x80, each increment requiring 2^22 system clock cycles or ~50 milliseconds. Hence, the watchdog times out after ~6.4 seconds. Every time the watchdog counter is reset by software, the counter returns to 0x00 and begins counting up again. The WDVALUE register (0x0400_0030) controls this reset value and hence, the time it takes the watchdog to expire.

Setting WDVALUE to 0x40 shortens the watchdog time-out from ~6.4 seconds to ~3.2 seconds. Setting it to 0x7f shortens the watchdog time-out to ~50 milliseconds.

The watchdog counter is cleared or reset by software by writing the WDREG_C register. This register is clearable only (write only) and has no bit definitions. The act of writing any value to its address has the effect of resetting the watchdog timer to the value specified by the WDVALUE register.

If the system is reset by the watchdog timer, this information is available after reset by reading the RESETCAUSE_S register (see the RESETCAUSE_S register definition later in this section).

5.2.3 busUnit

This section provides descriptions of each of the busUnit registers.

5.2.3.1 BUS_CHIPID Register

This register is read by software to determine the ID of the SOLO1 chip. All WebTV ASICS have this register defined at the same location.

Table 5-6 BUS_CHIPID Bit Definitions (0x0400_0000)

Bits	Symbol	Meaning
31:24	CHIPID[7:0]	Chip identification: SOLO1 = 0x03
23:20	CHIPREV[3:0]	Revision of silicon. Set to 0x0 initially and increments on each revision.
19:16	CHIPFAB[3:0]	Manufacturer of the silicon: Chip Express = 0x0 NEC = 0x1 Toshiba = 0x2
15:0	RESERVED	Reserved for future use.

5.2.3.2 BUS_CHPCNTL Register

This register controls various global chip functions, such as the reset watchdog timer, the audio clock dividers, and the time-out counts. This register is read/write for bits 17:0, while bits 31:30 are write-only. To write a value to any of the fields in this register, you must write all fields. In particular, this is important when enabling or disabling the watchdog counter.

Table 5-7 BUS_CHPCNTL Bit Definitions (0x0400_0004)

Bits	Symbol	Meaning
31:30	WDENAB	Writing a special sequence to this field enables/disables the reset watchdog timer in the busUnit. To enable the watchdog counter, software must write the following sequence: 00 -> 01 -> 10 -> 11. To disable the watchdog counter, software must write the following sequence: 11-> 10 -> 01 -> 00. After reset, this field reads back as 00. This watchdog timer resets the system if 64 VSYNC's have elapsed before software writes the BUS_WDREG to reset the counter.
29:18	RESERVED	Reserved for future use
17	SWPENDIAN	Causes SOLO1 to use the endianess that the CPU is not using. Should always be set to 0x0.

Table 5-7 BUS_CHPCNTL Bit Definitions (0x0400_0004) (Continued)

Bits	Symbol	Meaning
16	ALIASROM	Controls whether the ROM address space aliases onto the RAM address space. The reset value of this bit is 0x1, enabling the ROM alias. This functionality is provided for support of non-MIPS CPUs which may expect the reset vector to be at address 0x0000_0000 instead of 0x1fc0_0000. In the typical r4640 configuration, this bit will be set to 0x0 very soon after reset in order to make the RAM visible to the CPU.
15:0	TO- COUNT[15:0]	Time-out count value. The value in this field is compared against the time-out counter. When the time-out counter reaches this value, an error ack is returned to the CPU on reads. On reset, this field is set to 0xFFFF. These types of bus errors should only occur during reads from expansion bus devices when probing for peripherals after power-on.

5.2.3.3 BUS_INTSTAT Register

These registers contain the main interrupt status masked by the enable bits, and provide most of the status for any interrupt generated from SOLO1. Whenever an interrupt is detected, by the enable being set and a device interrupting, SOLO1 asserts the CPU_INT_N signal and keeps it asserted until the source of the interrupt is disabled. Its limit of six status bits and the fact that bits 1:0 are reserved as zero make it efficient to use a jump table to dispatch the next appropriate service routine.

Table 5-8 BUS_INTSTAT Bit Definitions (0x0400_0008, 0x0400_0050, 0x0400_0108)

Bits	Symbol	Meaning
31:8	RESERVED	Reserved for future use
7	None	Video interrupt status
6	None	Audio interrupt status
5	None	RIO (ISA) device interrupt status
4	None	SOLO1 device interrupt status
3	None	Timer interrupt status
2	None	Fence (error) interrupt status
1.0	RESERVED	Reserved for future use

5.2.3.4 BUS_INTEN Registers

These registers contain the main interrupt enables.

Table 5-9 BUS_INTEN Bit Definitions (0x0400_000C, 0x0400_010C)

Bits	Symbol	Meaning
31:8	RESERVED	Reserved for future use:
7	None	Video interrupt enable
6	None	Audio interrupt enable
5	None	RIO (ISA) device interrupt enable
4	None	SOLO1 device interrupt enable
3	None	Timer interrupt enable
2	None	Fence (error) interrupt enable
1:0	RESERVED	Reserved for future use

5.2.3.5 BUS_ERRSTAT Registers

This group of three registers is used to get the status of the most recent error that occurred in the system. The BUS_ERRSTAT (0x0400_0010) register stores the error status, BUS_ERRSTAT_S (0x0400_0054) sets the status bits, and BUS_ERRSTAT_C (0x0400_010) is used to clear the status bits.

Note that internal DMA transactions sent to a non-present expansion device will completely hang the system. It is up to software to ensure that this does not happen. If software detects a DMA transaction that has not completed after a ridiculously long time, it must reset the system via the watchdog timer. This sort of error is not detected or flagged in the hardware. Bus errors (timeouts) should only occur when probing for expansion devices after power-on.

Table 5-10 BUS_ERRSTAT Bit Definitions (0x0400_0010, 0x0400_0110, 0x0400_0054)

Bits	Symbol	Meaning
3159	RESERVED	Reserved for future use.
8	LOWREAD	When '1,' this bit indicates that the most recenterror ack was due to a low memory read fence violation. After reset, this bit is '0.' In order to enable another error, this bit is cleared, after detection by writing a '1' to the BUS_ERRSTAT_C.

Table 5-10 BUS_ERRSTAT Bit Definitions (0x0400_0010, 0x0400_0110, 0x0400_0054) (Continued)

Bits	Symbol	Meaning
7	LOWWRITE	When '1,' this bit indicates that the most recenterror ack was due to a low memory write fence violation. After reset, this bit is '0.' In order to enable another error, this bit is cleared, after detection by writing a '1' to the BUS_ERRSTAT_C.
6	F1READ	When '1,' this bit indicates that the most recent error ack was due to a read fence check violation between fence register set 1 taking place in the system. After reset, this bit is '0.' After detection, this bit must be cleared by writing a '1' to the BUS_ERRSTAT_C register in order to enable another error.
5	FIWRITE	When '1,' this bit indicates that the most recent error ack was due to a write fence check violation between fence register set 1 taking place in the system. After reset, this bit is '0.' After detection, this bit must be cleared by writing a '1' to the BUS_ERRSTAT_C register in order to enable another error.
4	F2READ	When '1,' this bit indicates that the most recent error ack was due to a read fence check violation between fence register set 2 taking place in the system. After reset, this bit is '0.' After detection, this bit must be cleared by writing a '1' to the BUS_ERRSTAT_C register in order to enable another error.
3	F2WRITE	When '1,' this bit indicates that the most recent error ack was due to a write fence check violation between fence register set 2 taking place in the system. After reset, this bit is '0.' After detection, this bit must be cleared by writing a '1' to the BUS_ERRSTAT_C register in order to enable another error.
2	TIMEOUT	When '1,' this bit indicates that the most recent error ack was due to a timeout taking place in the system. Read timeouts are synchronous, write timeouts are asynchronous. The write timeouts are flagged in conjunction with the BUS_INTSTAT register. After reset, this bit is '0.' After detection, this bit must be cleared by writing a '1' to the BUS_ERRSTAT_C register in order to enable another error.
1	RESERVED	Reserved for future use
0	OW	When '1,' this bit indicates that another error occurred before the first error could be serviced by the CPU.

5.2.3.6 BUS_ERREN Registers

This register pair is used to enable the various error-checking mechanisms in hardware. BUS_ERREN_S (at 0x0400_0014) contains the error-checking enable bits, and BUS_ERREN_C (at 0x0400_0114) is used to clear those bits.

Table 5-11 BUS_ERREN Bit Definitions (0x0400_0014, 0x0400_0114)

Bits	Symbol	Meaning
31.9	RESERVED	Reserved for future use.
8	LOWRDEN	When '1,' this bit indicates that low memory read errors are enabled. After reset, this bit is '0.'
7	LOWWREN	When '1,' this bit indicates that low memory write errors are enabled. After reset, this bit is '0.'
6	FIRDEN	When '1,' this bit indicates read errors on fence register set 1 are enabled. After reset, this bit is '0.'
5	F1WREN	When '1,' this bit indicates write errors on fence register set 1 are enabled. After reset, this bit is '0.'
4	F2RDEN	When '1,' this bit indicates read errors on fence register set 1 are enabled. After reset, this bit is '0.'
3	F2WREN	When '1,' this bit indicates write errors on fence register set 2 are enabled. After reset, this bit is '0.'
	TIMEEN	When '1,' this bit indicates time-outs on the bus are enabled. After reset, this bit is '0.'
10	RESERVED	Reserved for future use.

5.2.3.7 BUS_ERRADDR Register

This register is used to flag the address where an error was encountered.

Table 5-12 BUS_ERRADDR Bit Definitions (0x0400_0018)

Bits	Symbol	Meaning
31:0	ADDR[31:0]	This field captures the most recent address which caused the error flagged in the BUS_ERRSTAT register. After reset this field is undefined.

5.2.3.8 BUS_WDVALUE Register

This register controls the watchdog counter reset value. The watchdog time-out count is (64 - WDVALUE) VSYNCs.

Table 5-13 BUS_ WDVALUE Bit Definitions (0x0400_0030)

Bits	Symbol	Meaning
31:7	RESERVED	Reserved for future use.
6:0	WDVALUE	This field contains the reset value of the watchdog counter. After system reset, this field is 0x00, implying a watchdog time-out of 64 VSYNCs.

5.2.3.9 Fence Registers

The fence registers define the RAM space where valid write and read transactions can be performed. The fences either mark the uppermost address of the installed, physical memory, or delineate an area that is reserved for specific data.

5.2.3.9.1 BUS_LOWRDADDR Register

This register contains the upper bound of the low memory read protection fence. The lower bound is fixed at 0x0.

Table 5-14 BUS_LOWRDADDR Bit Definitions (0x0400_0034)

Bits	Symbol	Meaning
3111	RESERVED:	Prins field is hard coded to zeroes
10:2	ADDR	This field contains the address of the upper bound of the low memory read protection fence. If the fence is enabled, main memory read accesses are checked to see if their address is inside the low memory read fence. After reset, this field contains 0x000.
îo.	RESERVED'	Reserved for future use

5.2.3.9.2 BUS_LOWRDMASK Register

This register contains the mask-enable bits controlling which boundaries in main memory are checked against the low memory read fence.

Table 5-15 BUS_LOWRDMASK Bit Definitions (0x0400_0038)

Bits	Symbol	Meaning
31:17	RESERVED	Reserved for future use.
16:0	MASK	This field contains the mask bits that indicate which boundaries in main memory are checked against the low memory read fence.
		Bit Range
		0 0x0000_0000 -0x0000_0000 +BUS_LOWRDADDR 1 0x0010_0000 -0x0010_0000 +BUS_LOWRDADDR 2 0x0020_0000 -0x0020_0000 +BUS_LOWRDADDR
		16 0x00f0_0000 -0x00f0_0000 +BUS_LOWRDADDR
		After reset, this field contains 0x0000.

5.2.3.9.3 BUS_LOWWRADDR Register

This register contains the upper bound of the low memory read protection fence. The lower bound is fixed at 0x0.

Table 5-16 BUS_LOWRDADDR Bit Definitions (0x0400_003C)

Bits	Symbol	Meaning
31in	RESERVED	This field is hard coded to zeroes
10:2	ADDR	This field contains the address of the upper bound of the low memory write protection fence. If the fence is enabled, main memory write accesses are checked to see if their address is inside the low memory write fence. After reset, this field contains 0x000.
1:0.	RESERVED	Reserved for future use

5.2.3.9.4 BUS_LOWWRMASK Register

This register contains the mask enable bits controlling which boundaries in main memory are checked against the low memory write fence.

Table 5-17 BUS_LOWWRMASK Bit Definitions (0x0400_0040)

Bits	Symbol	Meaning
31:17	RESERVED	Reserved for future use
16:0	MASK	This field contains the mask bits that indicate which boundaries in main memory are checked against the low memory read fence. Bit Range
		0 0x0000_0000 -0x0000_0000 +BUS_LOWWRADDR 1 0x0010_0000 -0x0010_0000 +BUS_LOWWRADDR 2 0x0020_0000 -0x0020_0000 +BUS_LOWWRADDR
		16 0x00f0_0000 -0x00f0_0000 +BUS_LOWWRADDR
		After reset, this field contains 0x0000.

5.2.3.10 BUS_TCOUNT Register

This register is a timer that increments at a constant rate that is exactly half the CPU's maximum instruction issue rate. This register is read/write.

Table 5-18 BUS_ TCOUNT Bit Definitions (0x0400_0048)

Bits	Symbol	Meaning
31:0	Count	Value of the timer.

5.2.3.11 BUS_TCOMPARE Register

This register contains a value that is compared to the value contained in the TCOUNT register. The CPU generates a timer interrupt when the values match.

Table 5-19 BUS_ TCOMPARE Bit Definitions (0x0400_004C)

Bits	Symbol	Meaning
31:0	Compare	Value to be compared against the value in the TCOUNT register.

5.2.3.12 BUS_INTSTATRAW Register

The INTSTATRAW register can be used to poll interrupt events even when the interrupt is disabled and would not result in an actual CPU interrupt. This register contains the main interrupt status not masked by enable bits. This register is identical to the BUS_INTSTAT_S register.

Table 5-20 BUS_ INTSTATRAW Bit Definitions (0x0400_0050)

Bits	Symbol	Meaning
31:8	RESERVED	Reserved for future use
7	None	Video interrupt
6	None	Audio interrupt
5	None	RIO (ISA) device interrupt
4	None	SOLO1 device interrupt
3	None	Timer interrupt
2	None	Fence (error) interrupt
1 :0;	RESERVED	Reserved for future use

5.2.3.13 BUS_GPINTSTAT Registers

This group of three registers controls the status of the general-purpose I/O interrupts. GPINTSTAT (0x0400_0058) contains the interrupt status bits, GPINTSTAT_S (0x0400_0060) sets the bits, and GPINTSTAT_C (0x0400_0158) clears them.

Table 5-21 BUS_ GPINTSTAT Bit Definitions (0x0400_0058, 0x0400_0060, 0x0400_0158)

Bits	Symbol	Meaning
31:18	RESERVED	Reserved for future use (0)
17	None	GPIO 15 interrupt
16	None	GPIO 14 interrupt
15	None	GPIO 13 interrupt
14	None	GPIO 12 interrupt
13	None	GPIO 11 interrupt
12	None	GPIO 10 interrupt
11	None	GPIO 9 interrupt
10	None	GPIO 8 interrupt
9	None	GPIO 7 interrupt

Table 5-21 BUS_ GPINTSTAT Bit Definitions (0x0400_0058, 0x0400_0060, 0x0400_0158) (Continued)

Bits	Symbol	Meaning
8	None	GPIO 6 interrupt
7	None	GPIO 5 interrupt
6	None	GPIO 4 interrupt
5	None	GPIO 3 interrupt
4	None	GPIO 2 interrupt
3	None	GPIO 1 interrupt
2	None	GPIO 0 interrupt
1:0	RESERVED	Reserved for future use (0):

5.2.3.14 BUS_GPINTEN Registers

This pair of registers control the enable bits for the general-purpose I/O interrupts. BUS_GPINTEN_S ($0x0400_005C$) sets the enable bits, and BUS_GPINTEN_C ($0x0400_015C$) clears those bits.

Table 5-22 BUS_ GPINTPEN Bit Definitions (0x0400_005C, 0x0400_015C)

Bits	Symbol	Meaning
31:18	RESERVED	Reserved for future use (0).
17	None	GPIO 15 interrupt
16	None	GPIO 14 interrupt
15	None	GPIO 13 interrupt
14	None	GPIO 12 interrupt
13	None	GPIO 11 interrupt
12	None	GPIO 10 interrupt
11	None	GPIO 9 interrupt
10	None	GPIO 8 interrupt
9	None	GPIO 7 interrupt
8	None	GPIO 6 interrupt
7	None	GPIO 5 interrupt
6	None	GPIO 4 interrupt
5	None	GPIO 3 interrupt
4	None	GPIO 2 interrupt

Table 5-22 BUS_ GPINTPEN Bit Definitions (0x0400_005C, 0x0400_015C) (Continued)

Bits	Symbol	Meaning
3	None	GPIO 1 interrupt
2	None	GPIO 0 interrupt
1:0	RESERVED	Reserved for future use (0).

5.2.3.15 BUS_GPINTPOL Register

This register sets the polarity of the interrupts. A "1" is active high, and a "0" is active low.

Table 5-23 BUS_ GPINTPOL Bit Definitions (0x0400_0064)

Bits	Symbol	Meaning
31:18	RESERVED	Keserved for future use (0)
17	None	GPIO 15 interrupt
16	None	GPIO 14 interrupt
15	None	GPIO 13 interrupt
14	None	GPIO 12 interrupt
13	None	GPIO 11 interrupt
12	None	GPIO 10 interrupt
11	None	GPIO 9 interrupt
10	None	GPIO 8 interrupt
9	None	GPIO 7 interrupt
8	None	GPIO 6 interrupt
7	None	GPIO 5 interrupt
6	None	GPIO 4 interrupt
5	None	GPIO 3 interrupt
4	None	GPIO 2 interrupt
3	None	GPIO 1 interrupt
2	None	GPIO 0 interrupt
1:0	(RESERVED)	Reserved for future use (0).

5.2.3.16 BUS_AUDINTSTAT Register

This group of three registers controls the status of the audUnit interrupts. BUS_AUDINTSTAT (0x0400_0068) contains the interrupt status bits, BUS_AUDINTSTAT_S (0x0400_006C) sets the bits, and BUS_AUDINTSTAT_C (0x0400_0168) clears them.

Table 5-24 BUS_ AUDINTSTAT Bit Definitions (0x0400_0068, 0x0400_006C, 0x0400_0168)

Bits	Symbol	Meaning
31:7	RESERVED	Reserved for future ruse.
6	None	Soft modem DMA input interrupt
5	None	Soft modem DMA output interrupt
4	None	divUnit audio interrupt (Digital video)
3	None	Audio DMA input interrupt
2	None	Audio DMA output interrupt
1:0	RESERVED	Reserved for future use

5.2.3.17 BUS_AUDINTEN Registers

This pair of registers control the enable bits for the audUnit interrupts. BUS_AUDINTEN_S (0x0400_0070) sets the enable bits, and BUS_AUDINTEN_C (0x0400_0170) clears those bits.

Table 5-25 BUS_AUDINTEN Bit Definitions (0x0400_0070, 0x0400_0170)

Bits	Symbol	Meaning
31.7	- राज्ञधर्यक्षेत्र	Reserved for future use
6	None	Soft modem DMA input interrupt
5	None	Soft modem DMA output interrupt
4	None	divUnit audio interrupt (digital video)
3	None	Audio DMA input interrupt
2	None	Audio DMA output interrupt
1:0	RESERVED	Reserved for finture use

5.2.3.18 BUS_DEVINTSTAT Registers

This group of three registers controls the status of the devUnit interrupts. BUS_DEVINTSTAT (0x0400_0074) contains the interrupt status bits, BUS_DEVINTSTAT_S (0x0400_0078) is used to set the bits, and BUS_DEVINTSTAT_C (0x0400_0174) clears the bits.

Table 5-26 BUS_ DEVINTSTAT Bit Definitions (0x0400_0074, 0x0400_078, 0x0400_0174)

Bits	Symbol	Meaning
:31:8 ×	RESERVED	Reserved for future use
7	None	GPIO interrupt (masked by GPIO enables)
6	None	UART interrupt (sucUnit)
5	None	Smart Card interrupt (sucUnit)
4	None	Parallel Port interrupt (devUnit)
3	None	IR output interrupt (devUnit)
2	None	IR input interrupt (devUnit)
1:0	RESERVED	Reserved for future use (0).

5.2.3.19 BUS_DEVINTEN Registers

This pair of registers controls the devUnit interrupt enable bits. BUS_DEVINTEN_S (0x0400_007C) sets the bits, and BUS_DEVINTEN_C (0x0400_017C) clears them.

Table 5-27 BUS_ DEVINTEN Bit Definitions (0x0400_007C, 0x0400_017C)

Bits	Symbol	Meaning
318	RESERVED	Reserved for future use
7	None	GPIO interrupt (masked by GPIO enables)
6	None	UART interrupt (sucUnit)
5	None	Smart Card interrupt (sucUnit)
4	None	Parallel Port interrupt (devUnit)
3	None	IR output interrupt (devUnit)
2	None	IR input interrupt (devUnit)
1.0	RESERVED	Reserved for future use (0):

5.2.3.20 BUS_VIDINTSTAT Registers

This group of three registers controls the status of the vidUnit interrupts. BUS_VIDINTSTAT (0x0400_0080) contains the interrupt status bits, BUS_VIDINTSTAT_S (0x0400_0084) is used to set the bits, and BUS_VIDINTSTAT_C (0x0400_0180) clears the bits.

Table 5-28 BUS_ VIDINTSTAT Bit Definitions (0x0400_0080, 0x0400_084, 0x0400_0180)

Bits	Symbol	Meaning
31:6	RESERVED	Reserved for future use
5	None	divUnit interrupt (Digital video)
4	None	gfxUnit interrupt (GFX graphics core)
3	None	potUnit interrupt (Pixel output/timing)
2	None	vidUnit interrupt (Video unit)
1:0	RESERVED	Reserved for future use (0).

5.2.3.21 BUS_VIDINTEN Registers

This pair of registers controls the vidUnit interrupt enable bits. BUS_VIDVINTEN_S (0x0400_0088) sets the bits, and BUS_VIDINTEN_C (0x0400_0180) clears them.

Table 5-29 BUS_ VIDINTEN Bit Definitions (0x0400_0088, 0x0400_0188)

Bits	Symbol	Meaning
31.6	RESERVED	Reserved for future use
5	None	divUnit interrupt (Digital video)
4	None	gfxUnit interrupt (GFX graphics core)
3	None	potUnit interrupt (Pixel output/timing)
2	None	vidUnit interrupt (Video unit)
1:0	RESERVED	Reserved for future use (0).

5.2.3.22 BUS_RIOINTSTAT Registers

This group of three registers controls the status of the RIO bus interrupts. BUS_RIOINTSTAT (0x0400_008C) contains the interrupt status bits, BUS_RIOINTSTAT_S (0x0400_0090) is used to set the bits, and BUS_RIOINTSTAT_C (0x0400_018C) clears the bits.

Table 5-30 BUS_ RIOINTSTAT Bit Definitions (0x0400_008C, 0x0400_0090, 0x0400_018C)

Bits	Symbol	Meaning
31:6	RESERVED	Reserved for future use.
5	None	Device 3 interrupt
4	None	Device 2 interrupt
3	None	Device 1 interrupt (IDE hard disk by convention)
2	None	Device 0 interrupt (modem/ethernet by convention)
1:0	RESERVIED)	Reserved for future use.

5.2.3.23 BUS_RIOINTPOL Register

This register sets the polarity of the RIO bus devices. A "1" is active high, and a "0" is active low.

Table 5-31 BUS_ RIOINTPOL Bit Definitions (0x0400_0094)

Bits	Symbol	Meaning
31:6	RESHRVILE	Reserved for future use.
5	None	Device 3 interrupt
4	None	Device 2 interrupt
3	None	Device 1 interrupt (IDE hard disk by convention)
2	None	Device 0 interrupt (modem/ethernet by convention)
1:0	RESERVACE	Reserved for future use

The modem and ethernet controllers are believed to have negative polarity interrupts, while the IDE hard disk has a positive polarity interrupt. Based on this information, the RIOINTPOL register should be set to 0x08.

5.2.3.24 BUS_RIOINTEN Registers

This pair of registers controls the RIO bus interrupt enable bits. RIOVINTEN_S (0x0400_0098) sets the bits, and RIOINTEN_C (0x0400_0198) clears them.

Table 5-32 BUS_ RIOINTPEN Bit Definitions (0x0400_0098, 0x0400_0198)

Bits	Symbol	Meaning
31:6	RESERVED	Reserved for future use.
5	None	Device 3 interrupt
4	None	Device 2 interrupt
3	None	Device 1 interrupt (IDE hard disk by convention)
2	None	Device 0 interrupt (modem/ethernet by convention)
1:0	RESERVED	Reserved for future use

5.2.3.25 TIMINTSTAT Registers

This group of three registers controls the status of the timing interrupts. TIMINTSTAT (0x0400_009C) contains the interrupt status bits, TIMINTSTAT_S (0x0400_00A0) is used to set the bits, and TIMINTSTAT_C (0x0400_019C) clears the bits.

Table 5-33 BUS_ TIMINTSTAT Bit Definitions (0x0400_009C, 0x0400_0A0, 0x0400_019C)

Bits	Symbol	Meaning
31.4	RESHRVED)	Reserved for future use
3	None	System timer interrupt
2	None	Bus time-out (write bus error) interrupt
10	RESERVED	Reserved for future use (0).

5.2.3.26 TIMINTEN Registers

This pair of registers controls the timing interrupt enable bits. TIMVINTEN_S (0x0400_00A4) sets the bits, and TIMINTEN_C (0x0400_01A4) clears them.

Table 5-34 BUS_ TIMINTEN Bit Definitions (0x0400_00A4, 0x0400_01A4)

Bits	Symbol	Meaning
31:4	RESERVED	Reserved for future use
3	None	System timer interrupt
2	None	Bus time-out (write bus error) interrupt
1:0	RESERVED	Reserved for future use (0)

5.2.3.27 BUS_RESETCAUSE Registers

This pair of registers stores the cause of a reset. RESETCAUSE_S (0x0400_00A8) sets the bits, and RIOINTEN_C (0x0400_00AC) clears them.

Table 5-35 BUS_ RESETCAUSE Bit Definitions (0x0400_00a8, 0x0400_00AC)

Bits	Symbol	Meaning
31:3	RESERVED	Reserved for future use
2	SOFTWARE	Software caused reset
1	WATCHDOG	Watchdog caused reset
0	SWITCH	Reset switch caused reset

5.2.3.28 BUS_J1FENLADDR Register

This register contains the lower bound of the Java1 read/write protection fence.

Table 5-36 BUS_ J1FENLADDR Bit Definitions (0x0400_00B0)

Bits	Symbol	Meaning
31:26	RESERVED	Reserved for future use
25:5	ADDR	This field contains the address of the lower bound of the Java1 read/write protection fence. If the fence is enabled, main memory write accesses are checked to see if their address is inside the low memory write fence. After reset, this field contains 0x000.
40	RESERVED	Reserved for future use.

5.2.3.29 BUS_J1FENHADDR Register

This register contains the upper bound of the Java1 read/write protection fence.

Table 5-37 BUS_ J1FENHADDR Bit Definitions (0x0400_00B4)

Bits	Symbol	Meaning
31.26	RESERVED	Reserved for inture use
25:5	ADDR	This field contains the address of the upper bound of the Java1 read/write protection fence. If the fence is enabled, main memory write accesses are checked to see if their address is inside the low memory write fence. After reset, this field contains 0x000.
4:0	RESERVED.	Reserved for tuture use:

5.2.3.30 BUS_J2FENLADDR Register

This register contains the lower bound of the Java2 read/write protection fence.

Table 5-38 BUS_ J2FENLADDR Bit Definitions (0x0400_00B8)

Bits	Symbol	Meaning
31:26	RESERVED	Reserved for future use
25:5	ADDR	This field contains the address of the lower bound of the Java1 read/write protection fence. If the fence is enabled, main memory write accesses are checked to see if their address is inside the low memory write fence. After reset, this field contains 0x000.
4.0	RESERVED	Reserved for future use.

5.2.3.31 BUS_J2FENHADDR Register

This register contains the upper bound of the Java2 read/write protection fence.

Table 5-39 BUS_J2FENHADDR Bit Definitions (0x0400_00BC)

Bits	Symbol	Meaning
31-26	RESERVED	Reserved for future use:
25:5	ADDR	This field contains the address of the upper bound of the Java1 read/write protection fence. If the fence is enabled, main memory write accesses are checked to see if their address is inside the low memory write fence. After reset, this field contains 0x000.
40	RESERVED	Reserved for future use.

5.2.3.32 BUS_TOPOFRAM Register

This register contains the size of the physical RAM in megabytes.

Table 5-40 BUS_TOPOFRAM Bit Definitions (0x0400_00C0)

Bits	Symbol	Meaning
31:27	RESERVAD	Reserved for future use
26:20	ADDR	This field contains the size of the physical RAM in megabytes. A typical value is 0x0080_0000 (8MB address). After reset, this field contains 0x0400_0000 (64MB address).
19:0	RESERVED	Reserved for future use:

5.2.3.33 BUS_FENCECNTL Register

This register is used to cancel writes to the J1 and J2 valid address ranges.

Table 5-41 BUS_FENCECNTL Bit Definitions (0x0400_00C4)

Bits	Symbol	Meaning
31:5	RESERVED	Reserved for future use
4	None	When '1,' this bit cancels low writes. Reset value is '0.'
3	None	When '1,' this bit reverses the sense of J2. Reset value is '0.
2	None	When '1,' this bit cancels J2 writes. Reset value is '0.
1	None	When '1,' this bit reverses the sense of J1. Reset value is '0.
0	None	When '1,' this bit cancels J1 writes. Reset value is '0.

5.2.3.34 BOOTMODE Register

This register contains the first 25 bits of the CPUs reset string.

Table 5-42 BUS_BOOTMODE Bit Definitions (0x0400_00C8)

Bits	Symbol	Meaning
3125	RESERVED	Reserved for future use
24:0	None	First 25 bits of the CPU reset string.

5.2.3.35 USEBOOTMODE Register

This register enables the value in the BOOTMODE register to be used for the CPU reset configuration.

Table 5-43 BUS_ USEBOOTMODE Bit Definitions (0x0400_00CC)

Bits	Symbol	Meaning
31:1	RESERVED.	Reserved for future use
0	None	Use BOOTMODE register value.

5.2.3.36 BUS_WDREG_C Register

Software must write this register before 64 VSYNC's elapse, once the watchdog counter is enabled, or the system performs a hard reset.

Table 5-44 BUS_WDREG_C Bit Definitions (0x0400_0118)

Bits	Symbol	Meaning	
31:1	RESERVED	Reserved for future use.	
0	WDCLEAR	Writing a '1' to this field resets the watchdog counter. Writing a '0' has no effect.	

5.2.4 rioUnit Software Description

The rioUnit provides a shared interface to the system ROM, asynchronous devices (ISA type devices), and synchronous devices (WebTV Port devices). The CPU has access to all of the devices connected to the rioUnit. None of SOLO1's other logic units can access the rioUnit.

5.2.4.1 ROM Interface

There are two ROM banks, both of which can be accessed by the CPU.

- Bank 0 addresses: 0x1f00_0000 0x1f7f_ffff
- Bank 1 addresses: 0x1f80_0000 0x1fff_ffff

The physical (actual) ROM aliases through the memory space. Table 5-133 illustrates this aliasing for ROM banks with 2MB's of physical ROM each.

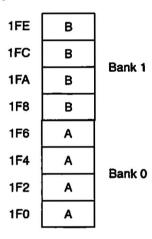


Figure 5-2 ROM Aliasing

Additionally, both ROM banks can be accessed through what is typically the RAM address space. This feature is provided for systems in which the CPU reset vector may reside in RAM space instead of ROM space.

ROM aliasing into RAM space is controlled by the BUS_CHPCNTL register (refer to the busUnit section, earlier in this chapter, for a description of this register). When ROM is aliased into RAM space, it is mapped so that the reset vector maps to address 0x0000_0000. ROM aliasing into RAM space is shown in Figure 5-3.

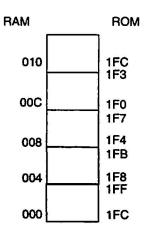


Figure 5-3 ROM Aliasing into RAM Space

Transaction characteristics for accesses to each ROM bank are set by the ROM Control registers (ROM_CNTL0, ROM_CNTL1).

The bit assignments for the ROM control registers (ROM_CNTL0 $(0\times0400_1004)$ and ROM_CNTL1 $(0\times0400_1008)$) are shown in the "rioUnit Registers" section that follows.

All bits, except write protect (bit 30), will be set at boot time and never changed. Their values are determined by the ROM timing specifications and the system clock speed, as follows:

```
INITWAIT = WETIME + 2
    WEDEL = 1
    WETIME = int ((rom access time + 23.37 ns) / cycle time)
- 2
    WETIME * cycle time >= 60 ns
```

The ROM access time is the CE_N to DATA time. This should be the same as the rated speed of the ROM (e.g. 90ns, 120ns, 150ns, etc.). The cycle time is the system bus cycle time. This should be calculated at boot time. For 83MHz, it should be ~12ns.

The Write Protect bit (bit 30) provides a way to protect ROM banks from unintended writes. When the write protect bit is set, any write to the protected ROM bank is cancelled and generates a bus error. Refer to the "busUnit Software Description" section, earlier in this chapter, for details on bus error behavior.

5.2.4.1.1 Typical System Configuration

By convention, Mask ROM is placed in Bank 1. The MIPS reset vector is 0x1fc0_0000, so the Mask ROM is constructed so that the first instruction executed is at 0x1fc0_0000. SOLO1-based systems are guaranteed to have a 32-bit wide Mask ROM bank.

Bank 0, if used, typically contains Flash ROM. SOLO1-based systems may have a 32-bit wide Flash ROM bank, although the typical configuration will be a 16-bit wide bank. The valid 16 bits may be either bits 31:16, or bits 15:0, depending upon the system architecture and the software's ability to deal with either configuration. When Bank 0 is read, any unconnected bits are guaranteed to return all 1's. If Bank 0 is less than 32-bits wide, code cannot be executed in place from Bank 0—only part of each instruction word would be valid.

5.2.4.2 Reset Behavior

All rioUnit interfaces are configured at reset to be slow (and hopefully safe for all potentially connected resources). The ROM interfaces are reset to an access time equal to 32 system clock cycles (for 66/83/100MHz clock, this is 480/384/320ns). The ROM access time can be shortened during the boot sequence to match the actual specifications of the parts used.

Device interfaces are also reset to slow timing, but are not critical at boot time, since no code is fetched from the devices. Timing that matches the actual system devices can be programmed during the boot sequence.

5.2.4.3 Asynchronous Device Interface

Asynchronous devices will probably be simple ISA device chipsets. Devices must be operated in slave mode: no DMA is provided. Only the CPU can access asynchronous (ISA) devices.

A maximum of eight asynchronous devices can be supported: four primary and four secondary devices. Device addressing is as follows:

Primary Devices

- Device 0 addresses: 0x1e00_0000 0x1e3f_ffff
- Device 1 addresses: 0x1e40_0000 0x1e7f_ffff
- Device 2 addresses: 0x1e80_0000 0x1ebf_ffff
- Device 3 addresses: 0x1ec0_0000 0x1eff_ffff

Secondary Devices

- Device 4 addresses: 0x1d00_0000 0x1d3f_ffff
- Device 5 addresses: 0x1d40_0000 0x1d7f_ffff
- Device 6 addresses: 0x1d80_0000 0x1dbf_ffff

Device 7 addresses: 0x1dc0_0000 - 0x1dff_ffff

All addresses are word-aligned, although supported devices use either an 8-bit or 16-bit data bus. If the device uses an 8-bit bus, these bits are connected to bits 7:0 of the rioUnit (RIO) data bus. Sixteen-bit devices are connected to bits 15:0 of the RIO bus. Bits 31:16 of the data bus are undefined for asynchronous device accesses. Byte, half-word and tri-byte accesses to these devices are not supported.

The bit assignments for the Device Control registers (DEV_CNTL0 (0x0400_100c), and DEV_CNTL1 (0x0400_1010)) are shown in the "rioUnit Registers" section that follows.

All bits will be set at boot time and never changed. Their values are determined by the device timing specifications and the system clock speed. Equations for typical system configurations are specified in the following subsections.

5.2.4.4 Typical System Configuration

Device 0

By convention, a hardware-based communications device is located in Device Slot 0. This would typically be an analog modem, or an ethernet controller. Register addresses for a modem would be as follows:

0x1e00_0000	Transmit/Receive buffer register
0x1e00_0004	Interrupt enable register
0x1e00_0008	FIFO control/Interrupt identify register
0x1e00_000c	Line Control register
0x1e00_0010	Modem control register
0x1e00_0014	Line Status register
0x1e00_0018	Modem Status register
0x1e00_001c	Scratch register

If the modem subsystem does not contain any ROM memory, the system is required to pre-load a modem ROM image into the SRAM, which replaces the ROM in the modem subsystem. If this is accomplished through a back door around the modem, then, while the system holds the modem in reset and enables the back door, the modem SRAM is accessible through the normal modem address space. The 128K SRAM is accessed through addresses 0x1e000000 - 0x1e03ffff.

The SRAM accesses must be word-aligned although the part is only 8-bits wide. The 8-bit SRAM data bus is connected to bits 7:0 of the RIO data bus. Using the modem back door to the SRAM may require a change in DEV_CNTL0 timing.

Typical device timing (DEV_CNTL0) for a Rockwell modem is as follows:

```
CERDYOFF = 6
STRRDYOFF = 4
CEOFF = STROFF + 2
STRON = 0
STROFF = int (72.44 / cycle time) + 1
```

Device 0 has a dedicated interrupt line which is connected to the interrupt pin on the modem or ethernet controller. See the busUnit description for more information on Device 0 interrupt enabling and status.

Device 1, Device 2

By convention, an IDE hard disk drive is connected to Device Slots 1 and 2. Register addresses for the hard drive would be as follows:

0x1e40_0000	Data register
0x1e40_0004	Error/Features register
0x1e40_0008	Sector Count register
0x1e40_000c	Sector Number register
0x1e40_0010	Cylinder Low register
0x1e40_0014	Cylinder High register
0x1e40_0018	Device/Head register
0x1e40_001c	Status register
0x1e80_0018	Alternate Status/Device Control register
0x1e80_001c	Drive Address register

The Device 1 dedicated interrupt line is connected to the interrupt signal from the hard drive. See the busUnit description for more information on Device 1 interrupt enabling and status.

A second IDE hard disk or other IDE device (like a CD-ROM drive) may be connected to secondary Device Slots 5 and 6.

Register addresses for a secondary IDE device would be as follows (registers for a CD-ROM drive may have different names and functions than hard disk registers):

0x1d40_0000	Data register
0x1d40_0004	Error/Features register
0x1d80_0018	Alternate Status/Device Control register
0x1d80_001c	Drive Address register

Typical device timing (DEV_CNTL1) for a Seagate Medalist hard disk drive is as follows:

```
CERDYOFF = 1

STRRDYOFF = 0

CEOFF = STROFF + 1

STRON = 1

STROFF = int (65 / cycle time) + 2
```

The Device 2 dedicated interrupt line is connected to the interrupt signal from the secondary IDE device. See the busUnit section for more information on Device 2 interrupt enabling and status.

Device 3

Device Slot 3 is currently unassigned. It may be used to support a keyboard controller, UART, or other standard-interface asynchronous device.

Secondary Devices 4-7

Secondary Device Slots 5 and 6 are assigned to a secondary IDE device (hard disk or CD-ROM). See the previous description for Devices 1 and 2. Secondary Device Slots 4 and 7 are currently unassigned.

5.2.4.5 Synchronous Device Interface

Synchronous devices are WebTV Port-compliant external or internal expansion devices. Devices must be operated in slave mode—no DMA is provided. The CPU and gfxUnit can access all synchronous (WebTV Port) devices. The device addresses are as follows:

Device 1 addresses	0x0480_0000 - 0x04ff_ffff
Device 2 addresses	0x0500_0000 - 0x057f_ffff
Device 3 addresses	0x0580_0000 - 0x05ff_ffff
Device 4 addresses	0x0600_0000 - 0x067f_ffff
Device 5 addresses	0x0680_0000 - 0x06ff_ffff
Device 6 addresses	0x0700_0000 - 0x077f_ffff
Device 7 addresses	0x0780_0000 - 0x07ff_ffff

Byte, half-word and tri-byte accesses are supported, although the target device must also support such accesses. All WebTV Port-compliant devices support a 32-bit data bus.

As a practical matter, SOLO1 only supports access to Devices 1-3. Devices 4-7 are usable, but only with specific (and potentially difficult) software workarounds which will not be discussed here. The WebTV Control register (WTV_CNTL (0x0400_101c)) is described in the "rioUnit Registers" section that follows.

All bits in the WebTV Port control register (WTV_CNTL (0x0400_101c)) are set at boot time and never changed. Their values are determined by system-level timing requirements, system clock speed and the supported expansion devices.

WebTV Port devices used to be managed by the memory (RAM) Unit and occupied the address space 0x0080_0000 - 0x03ff_ffff. Even though they have moved in the address map, their function remains the same (dependent on the particular device).

5.2.4.5.1 Typical System Configuration

Under most circumstances, all WebTV Port devices will be external expansion devices. The device installed in the main WebTV unit appears in Device Slot 1. The device installed in Device 1 appears in Device Slot 2, etc.

Refer to the Fido and WebTV Port interface specifications for more information about snooping the WebTV Port at boot time to determine which devices are connected.

The RIO Control register (RIO_CNTL (0x0400_1020)) determines the number of system cycles required to turn around the RIO port. The bits in this register will be set at boot time and never changed. Their values are determined by system and device timing requirements and system clock speed.

5.2.4.5.2 Typical System Configuration

Typical timing is as follows:

```
TURNAROUND = int (55 / cycle time)
TURNAROUND >= 1
```

5.2.5 rioUnit Registers

This section provides descriptions of each of the rioUnit registers.

5.2.5.1 RIO_SYSCONFIG Register

This register flags system configuration information for both software and hardware. Some of the bits program the SOLO1 chip directly. Most are used to software configure the hardware properly. This register contains the RIO_DATA[31:0] bus value, that is captured when reset goes away. Software is expected to figure out the system (CPU) clock frequency by using vertical interrupts and the NTSC bit.

Table 5-45 RIO_SYSCONFIG Bit Definitions (0x0400_1000)

Bits	Symbol	Meaning
31:23	RESERVED	Reserved for future use.
22	SMARTCARD1	When '1', this bit indicates that a Smart Card is present in slot 1.
21	SMARTCARD0	When '1', this bit indicates that a Smart Card is present in slot 0.
20	CPUTYPE	This bit indicates the type of CPU present: 1 = 4640 0 = 5230
19	CPUENDIAN	This bit sets the endianess of the CPU's system interface: 1 = big endian 0 = little endian
18	SYSTEM CONFIG	Th specific definition of this bit is system dependent. Its value is determined by the pullup/pulldown resistors on the system board. The SYSTEM CONFIG bit allows the system hardware to pass configuration information to the software.
17	CPUCLKMULT	This bit sets the CPU clock multiplier: 1 = 2x 0 = 3x
16:15	CPUDRIVE	This bit sets the bus speed: 11 = 83% 10 = 100% 01 = 50% 00 = 67%

Table 5-45 RIO_SYSCONFIG Bit Definitions (0x0400_1000) (Continued)

Bits	Symbol	Meaning
14:2	SYSTEM CONFIG	The specific definitions of these bits are system dependent. Their values are determined by the pullup/pulldown resistors on the system board. These SYSTEM CONFIG bits allows the system hardware to pass configuration information to the software.
1:0	RESERVED	Reserved for future use

5.2.5.2 ROM_CNTL0/1 Registers

ROM Control registers 0 and 1 are used to program the state machine that controls accesses to ROM Banks 0 and 1, respectively.

Table 5-46 ROM_CNTL0/1 Bit Definitions (0x0400_1004, 0x0400_1008)

Bits	Symbol	Meaning
31	RESERVED:	Reserved for future use.
30	PROTECT	Bank write protect (reset = 0x1)
20:16	INITWAIT	Access time in system cycles (reset = 0x1f)
15:10	RESERVED	Reserved for future use
9:8	WEDEL	Number of system cycles before WE_N asserted (reset = 0x3)
7.4	RESERVED	Reserved for future use
3:0	WETIME	Number of system cycles during which WE_N is asserted (reset = 0xf)

5.2.5.3 DEV_CNTL0/1 Registers

DEV_CNTL0 specifies timing for primary devices 0 and 3 and secondary devices 4 and 7. DEV_CNTL1 specifies timing for primary devices 1 and 2 and secondary devices 5 and 6. Figure 5-4 illustrates the timing required to hold the bus for a slow device. Note the STRRDYOFF and CERDYOFF signals.

Table 5-47 DEV_CNTL0-1 Bit Definitions (0x0400_100C, 0x0400_1010)

Bits	Symbol	Meaning
31:26	RESERVED	Reserved for future use
25:22	CERDYOFF	Number of system cycles after IOCHRDY before CE_N deasserts
21:18	STRRDYOFF	Number of system cycles after IOCHRD before IOR_N/IOW_N deasserts

Table 5-47 DEV_CNTL0-1 Bit Definitions (0x0400_100C, 0x0400_1010) (Continued)

Bits	Symbol	Meaning
17:12	CEOFF	Number of system cycles before CE_N deasserts (no IOCHRDY)
11: 6	STRON	Number of system cycles before IOR_N/IOW_N asserts
5: 0	STROFF	Number of system cycles before IOR_N/IOW_N deasserts (no IOCHRDY)

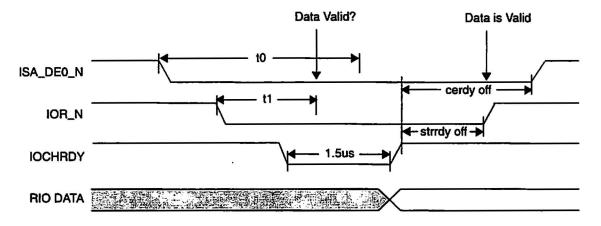


Figure 5-4 Hold the Bus!

5.2.5.4 WTV_CNTL Register

This register stores the clock divisor and the maximum number of outstanding transactions allowed on the WebTV Port.

Table 5-48 WTV_CNTL Bit Definitions (0x0400_101C)

Bits	Symbol	Meaning
31:8	RESERVED	Reserved for future use
7: 4	None	WebTV Port expansion clock divisor (reset = 0xf)
3: 0	None	Number of outstanding transactions allowed on the port (reset = $0x1$).

5.2.5.5 RIO_CNTL Register

This register stores the value of the RIO bus turnaround time. "Turnaround time" is defined as the period of time that must elapse following the completion of a bus transaction before a new transaction can be started.

Table 5-49 RIO_CNTL Bit Definitions (0x0400_1020)

Bits	Symbol	Meaning
31:4	RESERVED	Reserved for future use.
3: 0	TURNAROUND	Number of cycles required to turn around the RIO port (reset = 0xf).

5.2.6 audUnit Registers

DMA transfers are done from a logical block in memory, defined by a wordaligned start address. The output DMA buffer supports byte-aligned sizes. The input DMA buffer only supports 8-word aligned sizes.

Two sets of registers are maintained, one for the active buffer and one for the next buffer, which may not be valid. DMA transfers are started by loading a start address and buffer size into the next buffer registers, asserting the next buffer valid bit, and then asserting the DMA enable control bit.

The DMA engine transfers the next buffer information into the read—only registers of the active buffer and deasserts the next buffer valid bit. A counter begins counting up to the size of the active buffer and incrementing memory addresses as accesses are requested from the memory. The current state of the buffer is readable through the engines' control registers, including start address, buffer size, and the index of the last access. When the engine has completed the last access in the current buffer, it asserts an interrupt alerting the CPU that the buffer is now exhausted.

DMA chaining is implemented by both engines. When the current buffer has been exhausted, the engine checks the next buffer valid bit. If this bit is set, the next buffer information is transferred to the active buffer, the valid bit cleared, and transfers continue from the newly active buffer. This process continues until the DMA enable control bit is explicitly deasserted by the CPU, or until the active buffer is exhausted, and the next buffer valid bit is not set. In this condition, the engine automatically deasserts the enable bit and halts DMA transactions.

Both input and output DMA engines can be enabled via the AUD_IODMACNTL register, which allows software to simultaneously write both the AUD_IDMACNTL and AUD_ODMACNTL registers. This feature is useful for simultaneous start-up of the input and output channels.

5.2.6.1 AUD_OCSTART Register

This register is used to set and view the starting byte address of the currently executing output DMA transfer.

Table 5-50 AUD_OCSTART Bit Definitions (0x0400_2000

Bits	Symbol	Meaning
3126	RESERVED	Reserved for future use.
25:0	CADDR[25:0]	This field reflects the value of the starting address of the current DMA transaction.

5.2.6.2 AUD_OCSIZE Register

This register is used to set and view the size of the currently executing output DMA transfer. Note that audio DMA transfers can only be up to 64KB.

Table 5-51 AUD_OCSIZE Bit Definitions (0x0400_2004)

Bits	Symbol	Meaning
31:16	RESERVED	Reserved for future use.
15:0	CSIZE[15:0]	This field reflects the value of the size of the current DMA transaction.

5.2.6.3 AUD_OCCONFIG Register

This register is used to control the sample sizes and mono vs. stereo in the currently executing audio output DMA channel.

Table 5-52 AUD_OCCONFIG Bit Definitions (0x0400_2008)

Bits	Symbol	Meaning
31.2	RESERVED	Reserved for future use
1	8BIT	When '1,' this bit indicates the audio samples should be 8-bit. When '0,' 16-bit samples are used. After reset, this bit is undefined.
0	MONO	When '1,' this bit indicates mono samples should be used. When '0,' stereo samples are being used in the audio DMA channel. After reset, this bit is undefined.

5.2.6.4 AUD_OCCNT Register

This register is used to view the (byte) count of the currently executing output DMA transfer.

Table 5-53 AUD_OCCNT Bit Definitions (0x0400_200C)

Bits	Symbol	Meaning
31:16	RESERVED	Reserved for feture use
15:0	CCNT[15:0]	This field reflects the value of the index count of the current DMA transaction. Software can determine how far along the DMA transfer is by reading this register.

5.2.6.5 AUD_ONSTART Register

This register is used to read and write the starting address of the next output DMA transfer.

Table 5-54 AUD_ONSTART Bit Definitions (0x0400_2010)

Bits	Symbol	Meaning
30:26	RESERVED	Reserved for future use.
25:0	NADDR[25:0]	This field reflects the value of the starting address of the next DMA transaction.

5.2.6.6 AUD_ONSIZE Register

This register is used to read and write the size of the next output DMA transfer.

Table 5-55 AUD_ONSIZE Bit Definitions (0x0400_2014)

Bits	Symbol	Meaning
31:16	RESERVED	Reserved for future use
15:0	NSIZE[15:0]	This field reflects the value of the size of the next DMA transaction.

5.2.6.7 AUD_ONCONFIG Register

This register is used to control the sample sizes and mono vs. stereo in the "next" executing audio output DMA channel.

Table 5-56 AUD_ONCONFIG Bit Definitions (0x0400_2018)

Bits	Symbol	Meaning
812	RESERVED	Reserved for future tise
1	8BIT	When '1', this bit indicates the audio samples should be 8-bit. When '0', 16-bit samples are used. After reset, this bit is undefined.
0	MONO	When '1', this bit indicates mono samples should be used. When '0', stereo samples are being used in the audio DMA channel. After reset, this bit is undefined.

5.2.6.8 AUD_ODMACNTL Register

This register is used to control the audio output DMA channel.

Table 5-57 AUD_ODMACNTL Bit Definitions (0x0400_201C)

Bits	Symbol	Meaning
31:3	RESERVED -	Reserved for future use
2	DMAEN	When '1,' this bit indicates the DMA channel is enabled. When '0,' the DMA channel is disabled. After reset, this bit is '0.' A '1' to '0' transition on this bit flushes the current DMA transaction (i.e. the channel is reset).
1	NV	When '1,' this bit indicates that the NSTART and NSIZE registers are valid. Software should first set up the "next" registers, then enable the NV bit, and finally set the DMAEN bit in order to start the DMA channel.
0	NVF	When '1,' this bit indicates that the NV should stay valid forever. This allows software to have a continuous loop on a buffer, such as an audio sample.

5.2.6.9 AUD_ICSTART Register

This register is used to set and view the starting byte address of the currently executing input DMA transfer.

Table 5-58 AUD_ICSTART Bit Definitions (0x0400_2020)

Bits	Symbol	Meaning
3126	RESERVEDA	Reserved for future use
25:0	CADDR[25:0]	This field reflects the value of the starting address of the current DMA transaction.

5.2.6.10 AUD_ICSIZE Register

This register is used to set and view the size of the currently executing input DMA transfer. Note that audio DMA transfers can only be up to 64KB. Transfers must be 8-word aligned, so bits 5:0 are always zero.

Table 5-59 AUD_ICSIZE Bit Definitions (0x0400_2024)

Bits	Symbol	Meaning
31:16	RESERVED	Reserved for future use.
15:6	CSIZE[15:6]	This field reflects the value of the size of the current DMA transaction.
5:0	RESERVED	Always reads zero.

5.2.6.11 AUD_ICCNT Register

This register is used to view the (byte) count of the currently executing input DMA transfer.

Table 5-60 AUD_ICCNT Bit Definitions (0x0400_202C)

Bits	Symbol	Meaning
31:16	RESERVED	Reserved for future use:
15:6	CCNT[15:6]	This field reflects the value of the index count of the current DMA transaction. Software can determine how far along the DMA transfer is by reading this register.
5:0	RESERVED-	Always reads zero.

5.2.6.12 AUD_INSTART Register

This register is used to read and write the starting address of the next input DMA transfer.

Table 5-61 AUD_INSTART Bit Definitions (0x0400_2030)

Bits	Symbol	Meaning
30:26	RESERVED	Reserved for future use.
25:0	NADDR[25:0]	This field reflects the value of the starting address of the next DMA transaction.

5.2.6.13 AUD_INSIZE Register

This register is used to read and write the size of the next input DMA transfer.

Table 5-62 AUD_INSIZE Bit Definitions (0x0400_2034)

Bits	Symbol	Meaning
31:16	RESERVED	Reserved for future use.
15:6	NSIZE[15:6]	This field reflects the value of the size of the next DMA transaction.
5:0	RESERVED	Always reads zero. Writes are ignored.

5.2.6.14 AUD_IDMACNTL Register

This register is used to control the audio input DMA channel.

Table 5-63 AUD_IDMACNTL Bit Definitions (0x0400_203C)

Bits	Symbol	Meaning
313	MORNSHEER !	Reserved for future use
2	DMAEN	When '1,' this bit indicates the DMA channel is enabled. When '0,' the DMA channel is disabled. After reset, this bit is '0.' A '1' to '0' transition on this bit flushes the current DMA transaction (i.e. the channel is reset).
1	NV	When '1,' this bit indicates that the NSTART and NSIZE registers are valid. Software should first set up the "next" registers, then enable the NV bit, and finally set the DMAEN bit in order to start the DMA channel.
0	NVF	When '1,' this bit indicates that the NV should stay valid forever. This allows software to have a continuous loop on a buffer, such as an audio sample.

5.2.6.15 AUD_FCNTL Register

This register is used to control the clocks and data formats supporting various codecs.

Table 5-64 AUD_FCNTL Bit Definitions (0x0400_2040)

Bits	Symbol	Meaning
31:8	RESERVED	Reserved for future use:
7	DOUTLEFTJUST	If '1,' output data should be in left half of LRCLK phase. Default is '0' which selects right half.

 Table 5-64
 AUD_FCNTL Bit Definitions (0x0400_2040 (Continued))

Bits	Symbol	Meaning
6	DINRIGHTJUST	If '1,' input data should be in right half of LRCLK phase. Default is '0' which selects left half.
5	DOUTPOSBIT- CLK	If '1,' output data should be driven off of positive edge of BITCLK. Default is '0' which selects negative edge.
4	DINNEGBITCLK	If '1,' input data should be registered with the negative edge of BITCLK. Default is '0' which selects positive edge.
3	INVERTLRCLK	LRCLK sense should be inverted. 0 = LRCLK HI means left, LRCLK LO means right (default) 1 = LRCLK LO means left, LRCLK HI means right
2	INVERTBITCLK	BITCLK sense should be inverted. 0 = Negative edge of BITCLK coincident with LRCLK edge (default) 1 = Positive edge of BITCLK coincident with LRCLK edge
1:0	CODEC	Used to select different codec support: 00 = AKM AK4520 or Burr-Brown PCM3001 01 = AKM AK4532 10 = TITLC320AD50C 11 = Reserved

5.2.6.16 AUD_GPOCNTL Register

This register controls how the AUD_LRCLK, AUD_BITCLK, and AUD_SDATA signals are used as general-purpose output signals.

Table 5-65 AUD_GPOCNTL Bit Definitions (0x0400_2044)

Bits	Symbol	Meaning
31:6	RESERVED	Reserved for future use
5	SDATAEN	Enables AUD_SDATA to be a general-purpose output.
4	SDATAVAL	If SDATAEN is a '1,' the value of this bit is driven onto the AUD_SDATA pin.
3	BITCLKEN	Enables AUD_BITCLK to be a general-purpose output.

 Table 5-65
 AUD_GPOCNTL Bit Definitions (0x0400_2044) (Continued)

Bits	Symbol	Meaning
2	BITCLKVAL	If BITCLKEN is a '1,' the value of this bit is driven onto the AUD_BITCLK pin.
1	LRCLKEN	Enables AUD_LRCLK to be a general-purpose output.
0	LRCLKVAL	If LRCLKEN is a '1,' the value of this bit is driven onto the AUD_LRCLK pin.

5.2.6.17 AUD_IODMACNTL Register

This register is a shadow of the AUD_IDMACNTL and AUD_ODMACNTL registers, and ensures that both the input and output channels are controlled simultaneously. Changes to these register bits are reflected in the AUD_IDMACNTL and AUD_ODMACNTL registers and vice versa.

Table 5-66 AUD_IODMACNTL Bit Definitions (0x0400_205C)

Bits	Symbol	Meaning
317	RESERVED	Reserved for finture use
6	IN_DMAEN	Shadow of AUD_IDMACNTL (DMAEN)
5	IN_NV	Shadow of AUD_IDMACNTL (NV)
4	IN_IVF	Shadow of AUD_IDMACNTL (NVF)
3 ;	RESERVED	Reserved for future use.
2	OUT_DMAEN	Shadow of AUD_ODMACNTL (DMAEN)
1	OUT_NV	Shadow of AUD_ODMACNTL (NV)
0	OUT_NVF	Shadow of AUD_IDMACNTL (NVF)

5.2.7 vidUnit Registers

5.2.7.1 vidUnit Programming Overview

The vidUnit's main function is to supply pixel data from a memory-based frame buffer to the potUnit for display. It can utilize both interlaced and non-interlaced frame buffers.

There are several registers that control the various features of the vidUnit, and control the actual frame being displayed on the screen. Both the horizontal and vertical resolution of the video display are programmable. Figure 5-5 illustrates several vidUnit parameters that are key to understanding how the video display operates.

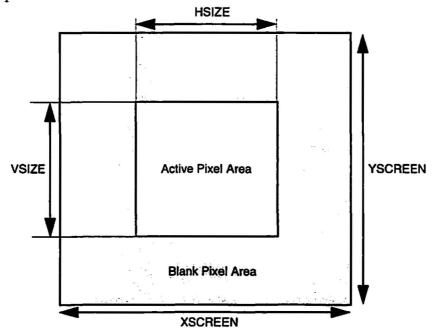
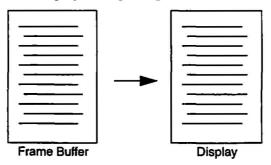


Figure 5-5 Video Display Overview

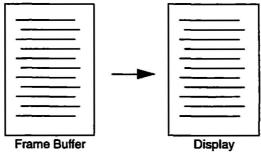
The vidUnit is simply a DMA engine that reads the frame buffer from main memory and presents it to the potUnit for display on the screen. The basic information that the engine needs is the size and starting location of the frame buffer. If the buffer needs to be scanned in an interlaced manner, the vidUnit also needs to know the width, in pixels, of a scan line, in order to skip every other line. As long as the DMA size is set exactly to the size of the active pixel area, the vidUnit continuously supplies the frame buffer for each field (for interlaced) or each frame (for non-interlaced).

There are four basic frame buffer vs. display formats that the vidUnit can handle:

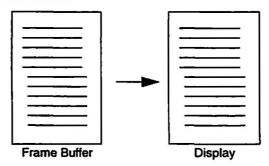
Integral frame buffer for progressive scan (no current application). In this
format, the frame buffer is read linearly, as a single frame, and passed to the
potUnit for display during a single frame time.



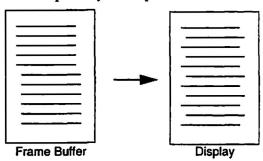
 Integral frame buffer for interlaced display (this is the general case for PAL and NTSC). In this case, every other line is read out and sent to the potUnit per field display. The first (odd) field begins on the first line of the frame buffer, the second (even) field begins on the second line. (For this model, there is no line zero.)



 Split frame buffer to progressive display (no current application). In this case, software or hardware have rendered each of the two fields into separate frame (really field) buffers. There is no way to re-integrate the fields in the SOLO1 hardware.



4. Split frame buffer to interlaced display (an alternative for NTSC and PAL). In this case, the vidUnit reads the frame buffer linearly, and presents it to the potUnit. However, since the fields are separated in the frame buffer, they will be presented separately to the potUnit.



In general, the potUnit should be programmed, but not enabled, before programming the vidUnit (the vidUnit gets its values for VSIZE and HSIZE from the potUnit). The vidUnit should then be programmed and enabled. Finally, the potUnit should be enabled. This allows the vidUnit pipeline to fill before the potUnit's first pixel request. In light of this, there is a definite sequence of events software must follow in order to correctly enable video. These rules are outlined in the following steps:

- 1. Set up the VSIZE and HSIZE registers in the potUnit. This will prevent undefined behavior from garbage values in those registers.
- Set up the DMA registers VID_NSTART and VID_NSIZE. NSIZE must be set to: (HSIZE*VSIZE)*2.
- Set the NV, NVF, and DMAEN bits of the VID_DMACNTL register. Ensure that
 the INTERLACEEN bit of this register is set to '0' for non-interlaced operation,
 or '1' for interlaced operation.
- 4. At this point, the vidUnit will load its pipeline and is ready for pixel requests from the potUnit.

5.2.7.2 Pixel Format

The pixel format within memory is shown in Table 5-67. Note that each pixel requires two bytes of storage. Thus, for a frame buffer that is H by V pixels, (HxVx2) bytes must be transferred. This also explains why all pixel counts and values must be multiples of two.

Table 5-67 Pixel Format in Memory

Bits	Symbol	Meaning
31:24	Y	Luminance value for pixel 'n.'
23:16	СЪ	Chrominance (blue) value for pixel 'n' and 'n+1.'
15:8	Y	Luminance value for pixel 'n+1.'
7:0	Cr	Chrominance (red) value for pixel 'n' and 'n+1.'

5.2.7.3 VID_CSTART Register

This register is used to set and view the starting word address of the currently executing video DMA transfer. It is loaded when the DMAEN and NV bits are both first set, or after the transfer of the last requested byte. It is read-only.

Table 5-68 VID_CSTART Bit Definitions (0x0400_3000)

Bits	Symbol	Meaning
31.26	RESERVED)	Reserved for future use
25:0	CADDR[25:0]	This field reflects the value of the starting address of the current DMA transaction.

5.2.7.4 VID_CSIZE Register

This register is used to view the size (in bytes) of the currently executing DMA transfer. It is loaded when the DMAEN and NV bits are both first set, or after the transfer of the last requested byte. This register is read-only.

Table 5-69 VID_CSIZE Bit Definitions (0x0400_3004)

Bits	Symbol	Meaning
E 521	REDERVED A	Reserved for future use
20:0	CSIZE[20:0]	This field reflects the value of the size (bytes) of the current DMA transaction.

5.2.7.5 VID CCNT Register

This register is used to view the (byte) count of the currently executing DMA transfer. It is cleared when the DMAEN and NV bits are both first set, or after the transfer of the last requested byte. It is incremented upon the receipt of a read acknowledgment from the memUnit. It is read-only.

Table 5-70 VID_CCNT Bit Definitions (0x0400_3008)

Bit	Symbol	Meaning
31:21	RESERVED	Reserved for future use.
20:0	CCNT[20:0]	This field reflects the value of the index count of the current DMA transaction. Software can determine how far along the DMA transfer is by reading this register.

5.2.7.6 VID_NSTART Register

This register is used to read and write the starting word address of the next DMA transfer. It is loaded into VID_CSTART when the DMAEN and NV bits are both first set, or after the transfer of the last requested byte.

Note: In SOLO1 Rev. 1.x, the starting address must be 32-byte aligned, i.e. NADDR[4:0] must be zero. In SOLO1 Rev 2.x, it need only be word-aligned, i.e. NADDR[1:0] must be zero.

Table 5-71 VID_NSTART Bit Definitions (0x0400_300C)

Bits	Symbol	Meaning
30:26	RESERVED	Reserved for friture use
25:0	NADDR[25:0]	This field reflects the value of the starting address of the next DMA transaction, i.e. the start of the frame buffer.

5.2.7.7 VID_NSIZE Register

This register is used to read and write the byte size of the next DMA transfer. It is loaded into VID_NSIZE when the DMAEN and NV bits are both first set, or after the transfer of the last requested byte.

Note: In SOLO1 Rev. 1.x, the size must be a multiple of 16 pixels (8 words), i.e. NSIZE[4:0] must be zero. In SOLO1 Rev. 2.x, the size need only be a multiple of two pixels, i.e. NSIZE[0] must be zero.

Table 5-72 VID_NSIZE Bit Definitions (0x0400_3010)

Bits	Symbol	Meaning
31:21	RESERVED	Reserved for future use.
20:0	NSIZE[20:0]	This field reflects the value of the size (in bytes) of the next DMA transaction. Note that the minimum video transfer is 16 words (64B).

5.2.7.8 VID_DMACNTL Register

This register is used to control the video DMA channel.)

Table 5-73 VID_DMACNTL Bit Definitions (0x0400_3014)

Bits	Symbol	Meaning
31:4	RESERVED	Reserved for future use
7:6	DIAGSEL	These bits select which of the internal signals are passed onto the diagnostic bus. For more information, please see the Verilog source for vidDMA. The settings of these bits have no visible effect for a normal system, but should be set to zero.
5	DVEPIXAVG	When '1,' the pixels sent to the dveUnit will have their chroma component averaged with the neighboring pixel. When '0', each pixel pair will have the exact same chroma values. This has no effect when using an external encoder. At reset, this bit is '0.'
4	VSYNCRESET	When '1' the vidDMA engine will be reset at every VSYNC, i.e. CSTART and CSIZE will be reloaded, CCNT cleared, and the vidUnit pipeline will be flushed and refilled. This prevents any persistent slippage. When '0,' the vidDMA engine will not be reset and will work exactly as in FIDO. This means that incorrect values of NSIZE or memory latency problems can cause persistent slippage. After reset, this bit is '0.'
3	DMA- INTERLACEEN	When '1,' this bit indicates that the DMA channel supports interlaced video, depending on the value of HSIZE and VSIZE. When '0,' the DMA channel supports non-interlaced mode. After reset this bit is '0.'

Table 5-73 VID_DMACNTL Bit Definitions (0x0400_3014 (Continued))

Bits	Symbol	Meaning
2	DMAEN	When '1,' this bit indicates the DMA channel is enabled. When '0,' the DMA channel is disabled. After reset, this bit is '0.' A '1' to '0' transition on this bit flushes the current DMA transaction (i.e. the channel is reset).
1	NV	Next Valid. When '1', this bit indicates the NSTART and NSIZE registers are valid. Software should first set up the "next" registers, then enable the NV bit and the DMAEN bit in order to start the DMA channel. After reset, this bit is '0.'
0	NVF	Next Valid Forever. When '1', this bit indicates the NV should stay valid forever. This allows software to have a continuous loop on a buffer such as a video frame. After reset, this bit is '0.'

5.2.7.9 VID_INTSTAT Register

This register pair controls the status of the vidUnit interrupts. VID_INSTAT_S sets the interrupt status, and induces an interrupt to the busUnit. VID_INSTAT_C clears the interrupt status. Either register may be read to determine the current status.

Table 5-74 VID_INTSTAT Bit Definitions (0x0400_3038, 0x0400_3138)

Bits	Symbol	Meaning
313	RESERVED	Reserved for future use
2	VIDDMA	This bit is set to '1' when the current video DMA channel completes, i.e. CCNT equals CSIZE.
1:0	RESERVED	Reserved for future rise

5.2.7.10 VID_INTEN Registers

This register pair provides the enable status of any video interrupt generated within SOLO1. Whenever an interrupt cause is detected when the enable is set, the corresponding bit in the INTSTAT register is set. VID_INTEN_S enables the catching of new vidUnit interrupt events. VID_INTEN_C disables the catching of new interrupts.

Table 5-75 VID_ INTEN Bit Definitions (0x0400_303C, 0x0400_313C)

Bits	Symbol	Meaning
31:7	RESERVED ***	Reserved for future use
2	VIDDMAEN	When '1,' this bit enables video DMA interrupts to pass through and be seen by the CPU. When '0,' this interrupt source is disabled. Software must write '1's to the VID_INTEN_S location (to set the bit), and to the VID_INTEN _C location (to clear the bit). This bit is set to '0' after reset.
1:0	RESERVED ==	Reserved for future use: These hit's will read back as 0

Note: The state of the enable bit has no direct effect on whether an interrupt is sent to the busUnit or not. If the bit in the INTSTAT bit is set, it is passed to the busUnit. This bit also has no effect on software's ability to set the INTSTAT bits by writing to the INTSTAT_S register.

5.2.7.11 VID_VDATA Register

This register reads the data on the VID_DATA pins. This is useful when the VID_DATA pins are configured as general-purpose inputs.

Table 5-76 VID_ INTEN Bit Definitions (0x0400_3040)

Bits	Symbol	Meaning
31.10°	RESIREVED .	Reserved for future use
9	None	Always '0.'
8	None	Always '0.'
7:0	VDATA	Value on the VID_DATA inputs.

5.2.7.12 Typical vidUnit Programming

This section provides an example of what typical programming for the vidUnit might look like:

At this point, the vidUnit will prime its internal FIFOs, and wait for the potUnit to start requesting pixels.

Setting the INTERLACE bit means that the DMA engine will skip every other line as it's reading. It determines the size of the line by looking at the POT_HSIZE register, located in the potUnit. Similarly, it determines the length of a field by examining the POT_VSIZE register, and dividing it by two. Behavior is undetermined when the VSize is odd, and will be supplied in an errata.

SOLO1 has no inherent timing interlocks between the different DMACNTL bits (i.e., you do not need to wait a certain number of cycles or fields between changing bits for correct functionality). However, there may be field timing issues to consider in order to correctly apply special effects.

At any time, if the DMA engine is idle, and the NV and DMAEN bits are both set, the engine will kick off. There are no other external gating factors (i.e., the unit doesn't wait for VSYNC or the completion of write-back operations), so care must be taken not to start the unit until the frame buffer (or at least the first line) is complete.

As soon as the engine kicks off, the NV bit is cleared unless the NVF bit is set. This is only checked immediately after kick-off. Changing the status of the NVF bit later will effect only subsequent transactions. Similarly, once the engine has started, clearing the NV bit will only prevent the engine from restarting. To halt the engine immediately, without waiting for the completion of the DMA transfer, clear the DMAEN bit (this bit works similarly to a software reset for the unit). In this way, the enable bit could be cleared, the unit's buffer size, location, and format information changed, and the engine restarted. Please note that there is no inherent interlock between the vidUnit and the potUnit. Turning off the vidUnit in the middle of a transfer can have unexpected visual consequences, and is not recommended.

The DVEPIXAVG bit enables averaging of the Chrominance values of the pixels being sent to the dveUnit. It has no effect on data heading to an external encoder, as most such encoders contain their own averaging hardware. Please note that the use of this averaging function requires higher video bandwidth, in order to have future pixels with which to create an average. There is the possibility of misaveraged pixels being undetected by the hardware slippage detector.

The VSYNCRESET bit instructs the DMA engine to terminate it's current transfer at the end of each frame, as determined by the potUnit. The can be used to resynchronize the vidUnit to the potUnit in case of slippage.

The value of the DIAGPICK bits are irrelevant for normal operation.

Note: The SOLO1 Rev. 2.x architecture will include a stride register, in order to implement the display of a subset of a frame buffer.

5.2.8 devUnit Registers

The devUnit registers support the IR unit, the GPIO lines, the parallel port, and other miscellaneous device interfaces.

5.2.8.0.1 DEV_IROLD Register

This register is used to read the input data from the external IR controller.

Table 5-77 DEV_IROLD Bit Definition (0x0400_4000)

Bits	Symbol	Meaning
31:0	IROLD[31:0]	This field contains the IR input data stream (32-bits).

5.2.8.1 DEV_LED Register

This register is used to control the three LEDs in the system.

Table 5-78 DEV_LED Bit Definition (0x0400_4004)

Bits	Symbol	Meaning
31:3	RESERVED	Reserved for future use
2	LED[2]	This bit controls the LED connected to the MISC_LED[2] signal on the chip. This LED indicates "power on." A '0' turns the LED on. A '1' turns the LED off. After reset, this bit is '1.'
1	LED[1]	This bit controls the LED connected to the MISC_LED[1] signal on the chip. This LED indicates "modem connected." A '0' turns the LED on. A '1' turns the LED off. After reset, this bit is '1.'
0	LED[0]	This bit controls the LED connected to the MISC_LED[0] signal on the chip. This LED indicates "modem connected." A '0' turns the LED on. A '1' turns the LED off. After reset, this bit is '1.'

5.2.8.2 DEV_IDCNTL Register

This register is used to control the unique ID chip in the system.)

Table 5-79 DEV_IDCNTL Bit Definition (0x0400_4008)

Bits	Symbol	Meaning
31:2	RESERVED	Reserved for future use.
1	IDDATAO	This bit is used to drive the ID data line. A '1' written to the bit reflects a '1' on the ID_DATA signal (open drain driver on chip), while a '0' reflects a '0' on the ID_DATA line. Software must make sure this bit is '1' when attempting to read the IDDATAI bit. After reset this bit is '1.'
0	IDDATAI	Software can read this bit to see what the "live" value of the ID_DATA signal is. Writes have no effect on this bit.

5.2.8.3 DEV_IICCNTL Register

This register is used to control the I²C bus.

Table 5-80 DEV_IICCNTL Bit Definition (0x0400_400C)

Bits	Symbol	Meaning
31.6	RESERVED.	Reserved for future use
5	IICCLKI	This field is used to read the value on the IIC_CLK signal. Writes to this field have no effect.
4	IICCLKOEN	A '1' allows IICCLKO to control the value being driven by SOLO1 onto the IIC_CLK signal. A '0' tristates the IIC_CLK signal. After reset this field is '0.'
3	IICCLKO	If IICCLKOEN is '1,' a '1' in this field tristates the IIC_CLK signal, allowing the external pullup to pull the signal high. A '0' causes SOLO1 to drive IIC_CLK low. After reset, this field is '0.'
2	IICDATAOEN	A '1' allows IICDATAO to control the value being driven by SOLO1 onto the IIC_DATA signal. A '0' tristates the IIC_DATA signal. After reset, this field is '0.'

Table 5-80 DEV_IICCNTL Bit Definition (0x0400_400C) (Continued)

Bits	Symbol	Meaning
1	IICDATAO	If IICDATAOEN is '1,' a '1' in this field tristates the IIC_DATA signal, allowing the external pullup to pull the signal high. A '0' causes SOLO1 to drive IIC_DATA low. After reset, this field is '0.'
0	IICDATAI	This field is used to read the value on the IIC_DATA signal. Writes to this field have no effect.

5.2.8.4 DEV_GPIOIN Register

This register contains the value on the GPIO pins, regardless of whether they are configured as inputs or outputs.

Table 5-81 DEV_GPIOIN Bit Definition (0x0400_4010)

Bits	Symbol	Meaning
31:16	RESERVED (1)	Reserved for future use
15:12	None	The value on GPIO[15:12].
11.	RESERVED	Reserved for future use
10:0	GPIOIN	The value on GPIO[10:0].

5.2.8.5 DEV_GPIOOUT Register

This register contains the value that is driven onto the GPIO bus, if the corresponding enable for each bit is set. Write a '1' to the set register (0x0400_4014) to set a bit. Write a '1' to the clear register (0x0400_4114) to clear a bit.

Table 5-82 DEV_GPIOOUT Bit Definition (0x0400_4014, 0x0400_4114)

Bits	Symbol	Meaning
31.8	RESERVED	Reserved for future use
15:12	None	The value that is driven onto GPIO[15:12], if the enable for each bit is set.
11	RESERVED	Reserved for future use
10:0	GPIOOUT	The value that is driven onto GPIO[10:0], if the enable for each bit is set.

5.2.8.6 DEV_GPIOEN Register

This register configures the GPIO lines as either outputs or inputs. Write a '1' to the set register (0x0400_4018) to set a bit. Write a '1' to the clear register (0x0400_4118) to clear a bit.

Table 5-83 DEV_GPIOEN Bit Definition (0x0400_4018, 0x0400_4118)

Bits	Symbol	Meaning
31:8	RESERVED	Reserved for future use.
15:12	None	If any of these bits are '0' the corresponding pins (GPIO[15:12]) are configured as outputs. Otherwise, the pins are inputs.
11	RESERVED	Reserved for future use
10:0	GPIOEN	If any of these bits are '0' the corresponding pins (GPIO[10:0]) are configured as outputs. Otherwise, the pins are inputs.

5.2.8.7 DEV_RAMDEL Register

This register must be programmed correctly for the gfxUnit to work properly. This register was created to provide some bits to program the RAM timing. For SOLO1, this register must be set to 0x55.

Table 5-84 DEV_RAMDEL Bit Definition (0x0400_4800)

Bits	Symbol	Meaning
31:8	RESERVED	Reserved for future use
7:6	RAMDELAYA0	Must be set to 1 to meet internal timing.
5:4	RAMDELAYA1	Must be set to 1 to meet internal timing.
3:2	RAMDELAYB0	Must be set to 1 to meet internal timing.
1:0	RAMDELAYB1	Must be set to 1 to meet internal timing.

5.2.8.8 DEV_IRIN_SAMPLE Register

This register controls the number of system clock cycles that occur between the internal IR receiver's sample clocks.

Table 5-85 DEV_IRIN_SAMPLE Bit Definition (0x0400_4020)

Bits	Symbol	Meaning
91:16	RESERVED	Reserved for future use
15:0	IRINSINT	Internal IR receiver sample interval.

5.2.8.9 DEV_IRIN_REJECT_INT Register

This register controls the window of time in which to reject glitches in the incoming IR signal.

Table 5-86 DEV_IRIN_REJECT_INT Bit Definition (0x0400_4024)

Bits	Symbol	Meaning
31:8	-RESERVED	Reserved for future use.
7:0	IRINRINT	Internal IR receiver rejection interval.

5.2.8.10 DEV_IRIN_TRANS_DATA Register

This register holds the oldest entry in the internal IR receiver FIFO.

Table 5-87 DEV_IRIN_TRANS_DATA Bit Definition (0x0400_4028)

Bits	Symbol	Meaning
91:16	RESERVED -	Reserved for future use
15	IRINSDATA	Value of the sampled data.
14:11	IRINWRPNT	Number of samples in the sample FIFO.
10:0	IRINSTIME	Number of sample clocks during which the value was sampled.

5.2.8.11 DEV_IRIN_STATCNTL Register

This register controls and checks the status of the internal IR receiver.

Table 5-88 DEV_IRIN_STATCNTL Bit Definition (0x0400_402C)

Bits	Symbol	Meaning
3138	RESERVED	Reserved for hithme use
7:4	IRININT- THRESH	Load the read FIFO with the number of entries needed to trigger an interrupt.
3	RESERVED	Reserved for future case
2	IRINDEBUG	If set, loops the IR receiver output back to the input.
1	IRINRESET	Reset the internal IR receiver state machines and FIFOs.
0	IRINENABLE	Enable the internal IR receiver: 0 = Use the external IR microcontroller (default) 1 = Use the internal IR receiver

5.2.8.12 IROUT Register Overview

The IR output signals consist of some number of pulses, whose size and rate are determined by a subcarrier. The subcarrier is either on or off, for any given pulse. An example of an IR output signal is shown in Figure 5-6.

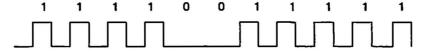


Figure 5-6 IR Output Signal

The IR FIFO's format is {count, bit}; where *count* is the number of repetitions minus one, and *bit* is either '1' (pulse present) or '0' (no pulse). The clock target is 12ns, so with typical subcarrier frequencies, there are approximately 2200 clocks per period.

5.2.8.13 DEV_IROUT_FIFO Register

This register stores the IR output data.

Table 5-89. DEV_IROUT_FIFO Bit Definition (0x0400_4040)

Bits	Symbol	Meaning
31:0	IROUT_FIFO	Contains the IR output data stream.

5.2.8.14 DEV_IROUT_STATUS Register

This register stores the IR status. Note that eleven of the fourteen bit descriptions in this register are defined by a pair of bits. With these bit pairs, writing a '1' to the high bit enables the lower bit. Explanations of the status bits are provided in the paragraphs following Table 5-90.

Table 5-90 DEV_IROUT_STATUS Bit Definitions (0x0400_4044)

Bits	Symbol	Meaning
31-29	RESERVED	Reserved for future use
28	ForceShift	0 = Normal operation 1 = Pop the data from the top of the FIFO
27:26	Subcarrier	10 = Operations are suspended 11 = Normal operation
25:24	Count	10 = Do not count periods 11 = Count periods
23:22	FIFO	10 = FIFO doesn't shift 11 = Normal operation

Table 5-90 DEV_IROUT_STATUS Bit Definitions (0x0400_4044) (Continued)

Bits	Symbol	Meaning
21:20	LoopMode	10 = Normal operation 11 = FIFO data recycles
19:18	ModulateDe- bugSignal	10 = On 11 = Off
17:16	CurrentBit	10 = No IR this period 11 = IR on
15:14	Polarity	10 = A '1' sends IR 11 = A '0' sends IR
13:12	OutputEnable	10 = Input mode 10 = Output mode
11:10	OutputBit	Current output bit after polarity adjustment. If Subcarrier on H signal is off, this value is meaningless.
9:8	Interrupt Enable	10 = No interrupts 11 = Interrupts enabled
7:6	InterruptBit	10 = No interrupt 11 = Interrupting
5:1	WriteAddress	The write address of the FIFO
0	InputValue	Data read from pin

The subcarrier, count, and FIFO bit pairs (Subcarrier, Count, and FIFO, respectively), stop their respective portions of the hardware. You can manually control the output by turning off the subcarrier and writing directly to the status register's output bit (Output bit).

There is a debug copy of the output bit (Output bit) that is sent to the IR input unit in order to test both the input and output logic. This copy can be sent without subcarrier modulation, which roughly approximates what is received from a normal IR receiver.

The Loop Mode bit (LoopMode) causes a FIFO entry to be written back into the FIFO after it is read out. This may be useful for different types of repeating signals.

If interrupts are enabled, they will only occur when there are three items or less in the FIFO.

5.2.8.15 DEV_IROUT_PERIOD Register

This register stores the clock period of the subcarrier. Note that the clock period is the bit value plus one.

Table 5-91 DEV_IROUT_PERIOD Bit Definition (0x0400_4048)

Bits	Symbol	Meaning
31:0	IROUT_PERIOD	The subcarrier's clock period.

5.2.8.16 DEV_IROUT_ON Register

This register stores the length of time that the IR LED is on during each subcarrier clock period.

Table 5-92 DEV_IROUT_ON Bit Definition (0x0400_404C)

Bits	Symbol	Meaning
31:0	IROUT_ON	The amount of time that the LED is on.

5.2.8.17 DEV_IROUT_CURRENT_PERIOD Register

This register stores the length of time remaining in the current subcarrier clock period.

Table 5-93 DEV_IROUT_CURRENT_PERIOD Bit Definition (0x0400_4050)

Bits	Symbol	Meaning
31:0	IROUT_CURRENT_PERIOD	The amount of time that the LED is on.

5.2.8.18 DEV_IROUT_CURRENT_ON Register

This register stores the length of time that the IR LED remains on.

Table 5-94 DEV_IROUT_CURRENT_ON Bit Definition (0x0400_4054)

Bits	Symbol	Meaning
31:0	IROUT_CURRENT_ON	The amount of time left for the LED to remain on.

5.2.8.19 DEV_IROUT_CURRENT_COUNT Register

This register stores the number of subcarrier periods remaining for the current action. Note that the number of periods remaining is the register bit value plus one.

Table 5-95 DEV_IROUT_CURRENT_COUNT Bit Definition (0x0400_4058)

Bits	Symbol	Meaning
31:0	IROUT_CURRENT_ COUNT	The amount of time left for the current action.

5.2.8.20 Parallel Port Registers

The parallel port registers support a 1284-compliant parallel port. SOLO1 hardware supports both Compatibility FIFO mode (the old handshake mechanism that only writes data out), and Extended Capabilities Port (ECP) mode, which is bi-directional. Other 1284-compliant modes can be supported in software.

5.2.8.21 DEV_PPORT_DATA Register

When in software-only mode (P_CNFG. Mode=0), this register is used both to directly control and examine the state of the parallel port data lines. When P_CTRL.Dir=1, writes to this register will change the state of the parallel port data pins, and when P_CTRL.Dir=0, reads of this register show the current state of the pins.

When in either hardware mode (P_CNFG. Mode = 1 or 2), writes to this register push the data byte onto a 16-byte FIFO. Data is then transferred to the printer via a hardware handshake. When in ECP Reverse mode, reads attempt to pull data off of the input FIFO, which is fed by the peripheral.

Table 5-96 DEV_PPORT_DATA Register (0x0400_4200)

Bits	Symbol	Meaning
31.8	RESERVED	Reserved for future use
7:0	PP_DATA	Parallel port data (read/write)

5.2.8.22 DEV_PPORT_CTRL Register

When in software-only mode (P_CNFG. Mode=0), this register is used to directly control the state of the parallel port control lines.

When in either hardware mode (Compatibility FIFO or ECP (P_CNFG. Mode = 1 or 2, respectively)), writes to this register will have no effect on the lines directly controlled by the hardware mode. Writes to the other fields will still control the state of the respective lines.

Table 5-97 DEV_PPORT_CTRL Register (0x0400_4204)

Bits	Symbol	Meaning
31.6	RESERVIED 3	Reserved for future use
5	PP_ECPCMND	This bit controls the state of the P_ATFD_N line during ECP mode (P_CNFG. Mode=2). Setting this bit to 1 will drive a 0 onto the P_ATFD_N line during ECP mode, indicating that the current data byte should be interpreted as an ECP Channel Select or RLE ByteCount command. Setting this bit to 0 will drive a 1 onto the line during ECP mode, indicating that the data byte simply contains data. No hardware mode overrides this bit, but the bit only has meaning during ECP mode (P_CNFG. Mode=2). Read/Write
4	PP_DIR	This bit controls the direction of the P_DATA lines. Setting this bit to '1' points data from SOLO1 to the printer, and setting it to '0' points data from the printer to SOLO1. No hardware mode overrides this bit. Read/Write
3	PP_SELECTIN_N	This bit controls the state of the P_SLIN_N pin. No hardware mode overrides this bit. Read/Write
2	PP_AUTOFD_N	This bit controls the state of the P_ATFD_N pin. This bit is overridden in ECP mode (P_CNFG.Mode=2), in which case the P_ATFD_N pin is controlled by the ECPCMND field described above. Read/Write

0

0

Table 5-97 DEV_PPORT_CTRL Register (0x0400_4204) (Continued)

Bits	Symbol	Meaning
1	PP_INIT_N	This bit directly controls the state of the P_INIT_N pin. when in software-only mode (P_CNFG.Mode=0), or in Compatibility FIFO mode (P_CNFG.Mode=1). This bit is overridden in ECP mode (P_CNFG.Mode=2), in which case the P_INIT_N pin is controlled by the hardware handshaking state machine. This bit determines in which direction the ECP hardware handshake operates: a '1' enables a 'forward' SOLO1-to-printer handshake; a '0' enables a 'reverse' printer-to-SOLO1 handshake. When in ECP mode, toggling this bit (1-0 or 0-1) flushes the FIFO. Read/Write
0	PP_STB_N	This bit directly controls the state of the P_STB_N pin when in software-only mode (P_CNFG . Mode = 0). This bit is overridden in Compatibility FIFO mode and ECP mode (P_CNFG . Mode = 1 or 2), in which case the P_STB_N pin is controlled by the hardware handshaking state machine. Read/Write

5.2.8.23 DEV_PPORT_STAT Register

This register is used to examine the state of the parallel port status lines.

Table 5-98 DEV_PPORT_STAT Register (0x0400_4208)

Bits	Symbol	Meaning
31:5	PRESERVABD.	Reserved for future use
4	PP_ACK_N	The state of the P_ERR_N pin. Read only
3	PP_SEL	The state of the P_SEL pin. Read only
2	PP_ERR	The state of the P_FLT_N pin. Read only
1	PP_FAULT_N	The state of the P_ACK_N pin. Read only
0	PP_BUSY	The state of the P_BUSY pin. Read only

5.2.8.24 DEV_PPORT_CNFG Register

This register is used to set the parallel interface in one of three different modes: Software-handshake mode, Compatibility FIFO mode, and ECP mode. Note that ECP mode must be pre-negotiated via software, with the interface in ECP forward idle phase, before turning ECP mode on with the P_CNFG register. Please refer to the IEEE 1284 specification for details on ECP mode negotiation.

Table 5-99 DEV_PPORT_CNFG Register (0x0400_420C)

Symbol	Meaning
RESERVED	Reserved for future use.
PP_MODE	This field controls the operating mode of the parallel port interface. Encoding is as follows: 00: Software-only mode 01: Compatibility FIFO mode 10: ECP mode 11: Reserved. Do not use. Read/Write
ì	ESERVED

5.2.8.25 DEV_PPORT_FIFOCTRL Register

This register is used to control FIFO-based interrupts generated by the parallel port and for resetting the FIFO.

 Table 5-100 DEV_PPORT_FIFOCTRL Register (0x0400_4210)

Bits	Symbol	Meaning
31:6	PREZERRAND)	Reserved for future vise
5	PP_FIFORESET	Writing a '1' to this field will reset the parallel port FIFO pointers. Valid entries inside the FIFO will be lost. Write only
4:0	PP_FIFOTHRESH	This field is only valid in a hardware-handshake mode (P_CNFG.Mode = 1 or 2), and controls when a FIFO-based interrupt is generated. When the available number of FIFO entries becomes equal to FifoThresh, an interrupt is generated if P_IEN.Thresh = 1. This interrupt will remain until P_CLRINT.Thresh is written. Note that an interrupt is only generated as the result of a hardware transfer between the devUnit and the device connected to the parallel port. Transfers between the devUnit and other SOLO1 units (i.e., software reads and writes) never generate an interrupt. Read/Write

5.2.8.26 DEV_PPORT_FIFOSTAT Register

This register is used to report FIFO-specific status, such as the number of available entries and FIFO overflow.

Table 5-101 DEV_PPORT_FIFOSTAT Register (0x0400_4214)

Bits	Symbol	Meaning
31:6	RESERVED	Reserved for future use.
5	PP_TIP	"Termination In Progress" This bit is set when the P_CNFG. Mode field has been set to 0, but the hardware interface hasn't completed the transfer of its current byte. When terminating from a hardware mode, software must first wait for this bit to be cleared before proceeding with Software-only mode. Read only
4:0	PP_FIFOAVAIL	This field reports the number of available FIFO entries. Read only

5.2.8.27 DEV_PPORT_TIMEOUT Register

This register is used to control the parallel port time-out threshold.

Table 5-102 DEV_PPORT_TIMEOUT Register (0x0400_4218)

Bits	Symbol	Meaning
311	RESERVED	Reserved for future use.
0	TIMEOUT	If an ECP Forward cycle has started, and the peripheral fails to complete the transfer, software can write a '1' to this bit to initiate a hardware abort sequence. This register is data strobe-sensitive. Writing a '0' has no effect. Write only

5.2.8.28 DEV_PPORT_STAT2 Register

The bits in this register correspond to the parallel port handshake signals. The value of each bit represents the level of the corresponding signal. This information is used to determine what state the handshaking state machine is in.

Table 5-103 DEV_PPORT_STAT2 Register (0x0400_421C)

Bits	Symbol	Meaning
314	RESERVED	Reserved for future use
3	PP_SELECTIN_N	Logical signal level (read only)
2	PP_AUTOFD_N	Logical signal level (read only)

Table 5-103 DEV_PPORT_STAT2 Register (0x0400_421C) (Continued)

Bits	Symbol	Meaning
1	PP_INIT_N	Logical signal level (read only)
0	PP_STB_N	Logical signal level (read only)

5.2.8.29 DEV_PPORT_IEN (Interrupt Enable) Register

This register is used to control which parallel port interrupt sources actually generate an interrupt.

Table 5-104 DEV_PPORT_IEN (Interrupt Enable) Register (0x0400_4220)

Bits	Symbol	Meaning
31:5	RESERVED	Reserved for future use.
4	PP_CFIFOEXCP	Allows the CFIFOEXCP interrupt source to generate an interrupt. See the P_IST register for a description of the source. Read/Write
3	PP_ACKDONE	Allows the ACKDONE interrupt source to generate an interrupt. See the P_IST register for a description of the source. Read/Write
2	PP_ECPREVERS E	Allows the ECPREVERSE interrupt source to generate an interrupt. See the P_IST register for a description of the source. Read/Write
1	RESERVED	Reserved for future use
0	PP_THRESH	Allows the THRESH interrupt source to generate an interrupt. See the P_IST register for a description of the source. Read/Write

5.2.8.30 DEV_PPORT_IST (Interrupt Status) Register

This register is used to report the state of the various interrupt sources, regardless of whether they are enabled via the P_IEN register.

Table 5-105 DEV_PPORT_IST (Interrupt Status) Register (0x0400_4224)

Bits	Symbol	Meaning
31:5	RESERVED	Reserved for future use
4	PP_CFIFOEXCP	Indicates that, while in Compatibility FIFO mode, the peripheral indicated an exception through either the assertion of P_ERROR or P_FLT_N. Read only

Table 5-105 DEV_PPORT_IST (Interrupt Status) Register (0x0400_4224) (Continued)

Bits	Symbol	Meaning
3	PP_ACKDONE	Indicates that, while in Software mode (Mode '0'), the peripheral has driven P_ACK_N from low to high. Useful for edge-detection with polling. Read only
2	PP_ECPREVERSE	Indicates that, while in Hardware-ECP mode, the peripheral is requesting a reverse transfer (by asserting P_FLT_N). Read only
1	RESERVED	Reserved for future use:
0	PP_THRESH	Indicates that, while in ECP or Compatibility FIFO modes, the number of free FIFO entries has reached the threshold value stored P_FIFOCTRL. Read only

5.2.8.31 DEV_PPORT_CLRINT Register

This register is used to clear interrupts. Writing a '1' to a given field will clear the associated interrupt source.

Table 5-106 DEV_PPORT_CLRINT Register (0x0400_4228)

Bits	Symbol	Meaning
31:5	RESERVED.	Reserved for future use
4	PP_CFIFOEXCP	Clears the CFIFOEXCP interrupt source. Write only
3	PP_ACKDONE	Clears the ACKDONE interrupt source. Write only
2	PP_ECPREVERSE	Clears the ECPREVERSE interrupt source. Write only
1	RESERVED :	Reserved for future use:
0	PP_THRESH	Clears the THRESH interrupt source. Write only

5.2.8.32 DEV_PPORT_ENABLE Register

This register enables/disables the parallel port.

Table 5-107 DEV_PPORT_ENABLE Register (0x0400_422C)

Bits	Symbol	Meaning
31:1	RESERVED	Reserved for future use.
0	PP_ENABLE	Writing a '1' causes the parallel port to function normally. Writing a '0' disables the port. Write only

5.2.8.33 DEV_DIAG Register

This register controls the externally-visible hardware debug bus, and which one of SOLO1's logic units has its diagnostic bits displayed. Debug data is driven onto a 6-bit bus. When the bus is enabled, the debug data appears on these external pins (see the following table).

Table 5-108 Debug Bus Data and the Corresponding External Pins

Debug Bit	External Pin Number
5	UART_DTR_N
4	SMC_PEN_N
3	SMC_RESET_N
2	SMC_CLK
1	SMC_INSERT_N
0	SMC_DATA

Table 5-109 DEV_DIAG Bit Definitions (0x0400_4804)

Bits	Symbol	Meaning
31:5	RESERVED	Reserved for future use.
4	DIAGEN	Enables debug bus data to appear on the external pins. Read/Write
3:0	DIAGSEL	This field selects which logic unit's debug bits are displayed on the bus (Read/Write): 0000 memUnit 0001 audUnit 0010 busUnit 0011 devUnit 0100 gfxUnit 0101 modUnit 0110 romUnit 0111 sucUnit 1000 Reserved 1001 Reserved 1011 Reserved 1011 Reserved 1100 divUnit 1101 dveUnit 1111 Reserved

Table 5-110 DEV_DEVDIAG Bit Definitions (0x0400_4808)

Bits	Symbol	Meaning
312	RESERVED.	Reserved for future use.
1:0	DEVDIAG	Selects which subunit of the devUnit logic has its debug data displayed on the debug bus (Read/Write): 00 IR Receiver 01 IR Transmitter 10 Parallel Port 11 Reserved

5.2.9 modUnit Registers

The modUnit is identical to the audUnit, except that only the 16-bit stereo format is supported, and only the right channel is used. Consequently, all sample data must be in the lower 16 bits of each sample word.

5.2.9.1 MOD_OCSTART Register

This register is used to set and view the starting byte address of the currently executing output DMA transfer.

Table 5-111 MOD_OCSTART Bit Definitions (0x0400_B000

Bits	Symbol	Meaning
31:26	RESERVED	Reserved for future use
25:0	CADDR[25:0]	This field reflects the value of the starting address of the current DMA transaction.

5.2.9.2 MOD_OCSIZE Register

This register is used to set and view the size of the currently executing output DMA transfer. Note that modem DMA transfers can only be up to 64KB.

Table 5-112 MOD_OCSIZE Bit Definitions (0x0400_B004)

Bits	Symbol	Meaning
31:16	RESERVED	Reserved for future use
15:0	CSIZE[15:0]	This field reflects the value of the size of the current DMA transaction.

5.2.9.3 MOD_OCCONFIG Register

This register can be written by software, but the results are undefined.

Table 5-113 MOD_OCCONFIG Bit Definitions (0x0400_B008)

Bits	Symbol	Meaning
31:0	RISSERVED.	Reserved for future use.

5.2.9.4 MOD_OCCNT Register

This register is used to view the (byte) count of the currently executing output DMA transfer.

Table 5-114 MOD_OCCNT Bit Definitions (0x0400_B00C)

Bits	Symbol	Meaning
31:16	RESERVED	Reserved for future use.
15:0	CCNT[15:0]	This field reflects the value of the index count of the current DMA transaction. Software can determine how far along the DMA transfer is by reading this register.

5.2.9.5 MOD_ONSTART Register

This register is used to read and write the starting address of the next output DMA transfer.

Table 5-115 MOD_ONSTART Bit Definitions (0x0400_B010)

Bits	Symbol	Meaning
30:26	RESERVED	Reserved for future use
25:0	NADDR[25:0]	This field reflects the value of the starting address of the next DMA transaction.

5.2.9.6 MOD_ONSIZE Register

This register is used to read and write the size of the next output DMA transfer.

Table 5-116 MOD_ONSIZE Bit Definitions (0x0400_B014)

Bits	Symbol	Meaning
31-16	MEGHKANHD)	Reserved for future use
15:0	NSIZE[15:0]	This field reflects the value of the size of the next DMA transaction.

5.2.9.7 MOD_ONCONFIG Register

This register is used to control the "next" executing modem output DMA channel.

Table 5-117 MOD_ONCONFIG Bit Definitions (0x0400_B018)

Bits	Symbol	Meaning
31:2	RESERVED	Reserved for future use.
1	8BIT	When '1', this bit indicates the audio samples should be 8-bit. When '0', 16-bit samples are used. After reset, this bit is undefined.
0	MONO	When '1', this bit indicates mono samples should be used. When '0', stereo samples are being used in the audio DMA channel. After reset, this bit is undefined.

5.2.9.8 MOD_ODMACNTL Register

This register is used to control the modem output DMA channel.

Table 5-118 MOD_ODMACNTL Bit Definitions (0x0400_B01C)

Bits	Symbol	Meaning
31.3	RESERVED	Reserved for future use
2	DMAEN	When '1,' this bit indicates the DMA channel is enabled. When '0,' the DMA channel is disabled. After reset, this bit is '0.' A '1' to '0' transition on this bit flushes the current DMA transaction (i.e. the channel is reset).
1	NV	When '1,' this bit indicates that the NSTART and NSIZE registers are valid. Software should first set up the "next" registers, then enable the NV bit, and finally set the DMAEN bit in order to start the DMA channel.
0	NVF	When '1,' this bit indicates that the NV should stay valid forever. This allows software to have a continuous loop on a buffer, such as a modem sample.

5.2.9.9 MOD_ICSTART Register

This register is used to set and view the starting byte address of the currently executing input DMA transfer.

Table 5-119 MOD_ICSTART Bit Definitions (0x0400_B020

Bits	Symbol	Meaning
31:26	RESERVED	Reserved for future use.
25:0	CADDR[25:0]	This field reflects the value of the starting address of the current DMA transaction.

5.2.9.10 MOD_ICSIZE Register

This register is used to set and view the size of the currently executing input DMA transfer. Note that modem DMA transfers can only be up to 64KB. Transfers must be 8-word aligned, so bits 5:0 are always zero.

Table 5-120 MOD_ICSIZE Bit Definitions (0x0400_B024)

Bits	Symbol	Meaning	
31:16	RESERVED	Reserved for future use.	
15:0	CSIZE[15:0]	This field reflects the value of the size of the current DMA transaction.	

5.2.9.11 MOD_ICCNT Register

This register is used to view the (byte) count of the currently executing input DMA transfer.

Table 5-121 MOD_ICCNT Bit Definitions (0x0400_B02C)

Bits	Symbol	Meaning	
317	RESERVED	Reserved for future use.	
15:6	CCNT[15:6]	This field reflects the value of the index count of the current DMA transaction. Software can determine how far along the DMA transfer is by reading this register.	
5:0=	RESERVED	Alwaysareads zero	

5.2.9.12 MOD_INSTART Register

This register is used to read and write the starting address of the next input DMA transfer.

Table 5-122 MOD_INSTART Bit Definitions (0x0400_B030)

Bits	Symbol	Meaning	
30:26	RESERVED -	Reserved for future use	
25:0	NADDR[25:0]	This field reflects the value of the starting address of the next DMA transaction.	

5.2.9.13 MOD_INSIZE Register

This register is used to read and write the size of the next input DMA transfer.

Table 5-123 MOD_INSIZE Bit Definitions (0x0400_B034)

Bits	Symbol	Meaning	
31:7	RESERVED	Reserved for future use	
15:6	NSIZE[15:6]	This field reflects the value of the size of the next DMA transaction.	
5:0	ROSSERVED	Always reads zero Writes are ignored	

5.2.9.14 MOD_IDMACNTL Register

This register is used to control the modem input DMA channel.

Table 5-124 MOD_IDMACNTL Bit Definitions (0x0400_B03C)

Bits	Symbol	Meaning	
313 -	RESERVED	Reserved for future use	
2	DMAEN	When '1,' this bit indicates the DMA channel is enabled. When '0,' the DMA channel is disabled. After reset, this bit is '0.' A '1' to '0' transition on this bit flushes the current DMA transaction (i.e. the channel is reset).	
1	NV	When '1,' this bit indicates that the NSTART and NSIZE registers are valid. Software should first set up the "next" registers, then enable the NV bit, and finally set the DMAEN bit in order to start the DMA channel.	
0	NVF	When '1,' this bit indicates that the NV should stay valid forever. This allows software to have a continuous loop on a buffer, such as an audio sample.	

5.2.9.15 MOD_FCNTL Register

This register is used to control the clocks and data formats supporting various codecs.

Table 5-125 MOD_FCNTL Bit Definitions (0x0400_B040)

Bits	Symbol	Meaning	
31:8	RESERVED	Reserved for future use.	
7	DOUTLEFTJUST	If '1,' output data should be in left half of LRCLK phase. Default is '0' which selects right half.	
6	DINRIGHTJUST	If '1,' input data should be in right half of LRCLK phase. Default is '0' which selects left half.	
5	DOUTPOSBIT- CLK	If '1,' output data should be driven off of positive edge of BITCLK. Default is '0' which selects negative edge.	
4	DINNEGBITCLK	If '1,' input data should be registered with the negative edge of BITCLK. Default is '0' which selects positive edge.	
3	INVERTLRCLK	LRCLK sense should be inverted. 0 = LRCLK HI means left, LRCLK LO means right (default) 1 = LRCLK LO means left, LRCLK HI means right	
2	INVERTBITCLK	BITCLK sense should be inverted. 0 = Negative edge of BITCLK coincident with LRCLK edge (default) 1 = Positive edge of BITCLK coincident with LRCLK edge	
1:0	CODEC	Used to select different codec support: 00 = AKM AK4520 or Burr-Brown PCM3001 01 = AKM AK4532 10 = TTTLC320AD50C 11 = Reserved	

5.2.9.16 MOD_GPOCNTL Register

This register controls how the MOD_LRCLK, MOD_BITCLK, and MOD_SDATA signals are used as general-purpose output signals.

Table 5-126 MOD_GPOCNTL Bit Definitions (0x0400_B044)

Bits	Symbol	Meaning	
31:6	RESERVED	Reserved for future use	
5	SDATAEN	Enables MOD_SDATA to be a general-purpose output.	
4	SDATAVAL	If SDATAEN is a '1,' the value of this bit is driven onto the MOD_SDATA pin.	
3	BITCLKEN	Enables MOD_BITCLK to be a general-purpose output.	
2	BITCLKVAL	If BITCLKEN is a '1,' the value of this bit is driven onto the MOD_BITCLK pin.	
1	LRCLKEN	Enables MOD_LRCLK to be a general-purpose output.	
0	LRCLKVAL	If LRCLKEN is a '1,' the value of this bit is driven onto the MOD_LRCLK pin.	

5.2.9.17 MOD_IODMACNTL Register

This register is a shadow of the MOD_IDMACNTL and MOD_ODMACNTL registers, and ensures that both the input and output channels are enabled simultaneously. Changes to these register bits are reflected in the MOD_IDMACNTL and MOD_ODMACNTL registers and vice versa.

Table 5-127 MOD_IODMACNTL Bit Definitions (0x0400_B05C)

Bits	Symbol	Meaning	
91.7	RESERVED :	Reserved for future use.	
6	IN_DMAEN	Shadow of MOD_IDMACNTL (DMAEN)	
5	IN_NV	Shadow of MOD_IDMACNTL (NV)	
4	IN_IVF	Shadow of MOD_IDMACNTL (NVF)	
3.	RESERVED	Reserved for future rise	
2	OUT_DMAEN	Shadow of MOD_ODMACNTL (DMAEN)	
1	OUT_NV	Shadow of MOD_ODMACNTL (NV)	
0	OUT_NVF	Shadow of MOD_IDMACNTL (NVF)	

5.2.10 memUnit Registers

The memUnit provides the interface to main memory for the CPU and for all of the SOLO1 DMA engines. There are four registers used to control the various aspects of the memUnit operation:

- Mem_Timing Register
- Mem_Burp Register
- MEM_REFCNTRL Register
- MEM_CMD Register

5.2.10.1 MEM_TIMING Register

This register maintains the timing parameters required by the memUnit to access the SDRAMs. The register resets to values designed to work with 100MHz SDRAMs when SOLO1 is operating at 83.3MHz (CAS latency = 3). Table 5-129 lists the values that must be written to the Mem_Timing register for various types of SDRAM and their frequencies.

Table 5-128 System Frequency vs. SDRAM Speed

SDRAM	System Frequency	
Speed	83.3MHz	66.7Mhz
83MHz	Invalid	8E48E004*
100MHz	8E48E004*	DBFB2004
125MHz	8E48E004*	DBFB2004

^{*} Indicates the values present following reset.

Table 5-129 provides the individual bit definitions for the Mem_Timing register.

Table 5-129 MEM_TIMING Bit Definition (0x0400_5000)

Bits	Symbol	CASLatency Values (CL3) (83.3MHz)	CASLatency Values (CL2) (66.7Mhz)
31:30	TCAC	3 (Reset value)	2
·29 :	RESERVED		
28:26	TRDRP	4 (Reset value)	3
25:24	TWR	2 (Reset value)	2
23:22	TRCD	3 (Reset value)	2
21:20	TRP	3 (Reset value)	2

Table 5-129 MEM_TIMING Bit Definition (0x0400_5000) (Continued)

Bits	Symbol	CASLatency Values (CL3) (83.3MHz)	CASLatency Values (CL2) (66.7Mhz)
19:17	TRAS	5 (Reset value)	4
16:13	TRC	9 (Reset value)	7
12:4	RESERVED		
3:0	BurstSize	4 (Reset value)	4

5.2.10.2 MEM_BURP Register

The MEM_BURP register is used to control the memory access arbitration algorithm. Essentially, there are two fields: The MEM_BURPCT and MEM_BURPEN. The former is a count value, the latter is used as a mask to either include or exclude a device from the burp calculation.

Note: MEM_BURPCT must always contain a non-zero value.

The purpose of the burp calculation is to guarantee real-time response from the memUnit for real-time oriented devices (video readout, audio, and digital video write) while also allowing the gfxUnit as much undivided access as possible.

The algorithm is as follows: When the gfxUnit is accessing memory in word mode, it may do so forever. However, when a burp-enabled device asserts a request, the burpcount is enabled. When the burp count reaches that which is programmed in this register, the gfxUnit is kicked off, so the higher-priority requests can be serviced.

Table 5-130 MEM_BURP Bit Definition (0x0400_5008)

Bits	Symbol	Meaning	
31	MEMBUSBEN	R/W Reset to 1 Enables the busUnit to participate in the burp algorithm	
30	MEMVIDBEN	R/W Reset to 1 Enables the vidUnit to participate in the burp algorithm	
29	MEMAUDBEN	R/W Reset to 1 Enables the audUnit to participate in the burp algorithm	

Table 5-130 MEM_BURP Bit Definition (0x0400_5008) (Continued)

Bits	Symbol	Meaning
28	MEMMODBEN	R/W Reset to 1 Enables the modUnit to participate in the burp algorithm
27	MEMDIVBEN	R/W Reset to 1 Enables the divUnit to participate in the burp algorithm
26.	RESERVED	Reserved for future use
25	MEMGWBBEN	R/W Reset to 1 Enables graphics write-back unit to participate in the burp algorithm
24	MEMREFBEN	R/W Reset to 1 Enables refresh to participate in the burp algorithm
23:8	RESERVED	Reserved for future use: Read as Zero
7:0	MEMBURPCT	RW Reset to 0xFF The number of cycles before the gfxUnit is kicked off the memory controller. Do not program to '0' or gfxUnit performance will be slowed by a factor of ten.
		If the system frequency is 83.3MHz, this field should be 0xFF00000A. If the system frequency is 66.7MHz, this field should be 0xFF000008.

5.2.10.3 MEM_REFCTRL Register

This register is used to control how the SDRAMS are refreshed, and when and if they enter auto-refresh mode. This register must be programmed upon initialization in order to enable refresh to commence and to enable the refresh count.

Table 5-131 MEM_REFCTRL Bit Definition (0x0400_500C)

Bit	Symbol	Meaning
31	RefEnb	R/W if SetRefEnb is enabled. Reset to 0 If '1,' memory is refreshed at the interval specified by RefCount.
30	SetRefEnb	Write Only, Read as 0. Software writes a '1' to this bit to allow refEnable to be set. Writing a '0' to this bit clears RefEnb.
29	AutoRefEnb	R/W if SetAutoRefEnb is enabled. Reset to '0.' If '1,' the memory controller will place the SDRAMs in auto-refresh mode, if it receives no requests between refresh intervals. When a request is present and the SDRAMs are in auto-refresh mode, the memory controller returns to refresh mode, and services the request. Note that SetAutoRefEnb only works with Toshiba SDRAMs.
28	SetAutoRefEnb	Write Only, Read as 0. SW writes a '1' to this bit to allow AutoRefEnable to be set. Writing a '0' to this bit clears AutoRefEnb.
27:11	RESERVED	Reserved for ruture use
10:0	RefCount	R/W Reset to '0.' SW writes this field to determine the number of clock cycles between refresh. There are 2048 rows, each refreshed within 32ms. 66.7MHz: 0xC0000411 83.3 MHz: 0xC0000516

5.2.10.4 MEM_CMD Register

IMPORTANT—Before any commands can be given to the SDRAMS via this register, refresh must be disabled. Failure to do so will have uncertain results.

Also, note that a read of this register will indicate the last command that was executed on the SDRAMs. This includes refresh cycles and auto-refresh enter and exit cycles that were generated automatically.

This register is used to perform specific operations on the SDRAM chips, such as Mode Register Set and other functions involved in initializing the SDRAMs. At system power-on, or SOLO1 reset, software must program this register in order for memory to operate properly. The required initialization sequence follows:

- 1. Write 0xa600_0000 to MEM_CMD register (PD_EXIT enables the SDRAM clock and the number-two banks of SDRAM).
- Write 0x8000_003A to MEM_CMD (Precharge All).
- Write 0x2000_002A to MEM_CMD (Mode Register Set for CL 2—66.2MHz).
- If the system frequency is 66.7MHz, write 0x2000_003A to MEM_CMD (Mode Register Set for CL 3—83.3MHz). Else, write 0x4800_0000 to MEM_CMD (Refresh all chips).
- 5. Write 0x4800_0000 to MEM_CMD (Refresh all chips).
- 10. Write 0x4800_0000 to MEM_CMD (Refresh all chips).
- 11. Write 0x4800_0000 to MEM_CMD (Refresh all chips).
- If the system frequency is 66.7MHz, write 0xC000_0411 to MEM_REFCTRL (Turn on Refresh), else write 0xC000_0516.
- 13. If 66.2MHz, write 8E48E004 to the MEM_TIMING register.
- 14. Determine the memory configuration and program the memory size.

Table 5-132 MEM_CMD Bit Definition (0x0400_5010)

Bits	Symbol	Meaning
31:28	MemCmd[3:0]	This field is used to send a command to the SGRAM part(s). Table 5-133 shows the encoding. This field is write-only.
27:26	MemSize	00: One load per bit on the data bus. 4Mbytes= Two 1M-bit x16-bit SDRAMs 8Mbytes= Four 2M-bit x 8-bit SDRAMs
		01: Two loads per bit on the data bus. 8Mbytes= Four 16M-bit x 16-bit SDRAMs 16Mbytes= Eight 2M-bit x 8-bit SDRAMs
		10: One load per bit on the data bus. 16Mbytes= Two 4M-bit x 16-bit SDRAMs 32Mbytes= Four 8M-bit x 8-bit SDRAMs 64Mbytes= Eight 16M-bit x 4-bit SDRAMs
		Reset to 00.
25	MemSizeSet Enable	If '1,' MemSize changes to whatever value is in the MemSize field. If '0,' MemSize field is cleared.
24.11	RESERVED. *	Reserved for future use
10:0	MemMSR	Program to " $3A$ " for $CL = 3$ (Reset value), and to " $2A$ " for $CL = 2$.

Table 5-133 MEMCMD Encoding

MEMCMD[3:0]	OPCODE	Encoding	
0x0 = ; = ;	RRESERVED	Reserved for auture use.	
0x1	RESERVED	Reserved for intime use	
0x2	MRS	Mode Register Set	
0x3	RESERVED	Reserved for uture use	
0x4	REF	Refresh	
0.65;	RESERVED	Reserved for tuture use	
0x6	RESERVED	Reserved for future use	
0 07	RESERVED	Reserved for future use.	
0x8	PALL	Precharge All Banks	

Table 5-133 MEMCMD Encoding (Continued)

MEMCMD[3:0]	OPCODE	Encoding
0x9	PD_ENTRY	Power-down Entry
0xa	PD_EXIT	Power-down Exit
Oxb	RESERVED	Reserved for future use.
0xc	RESERVED	Reserved for future use.
0xd	SELF_ENTRY	Auto-refresh power-down Entry
0xe	SELF_EXIT	Auto-refresh power-down Exit
0xf	RESERVED	Reserved for future use.

5.2.11 gfxUnit Registers

This section describes all of the gfxUnit registers.

5.2.11.1 GFX_CONTROL Register

This register is used to configure the gfxUnit.

Table 5-134 GFX_CONTROL Bit Definition (0x0400_6004)

Bits	Symbol	Meaning
31-8	RESERVED	Reserved for future use:
7	gfxEn	0 = Unit is forced idle 1 = Registers are valid and processing should begin/continue
6	DeltaTim	0 = Compatibility mode 1 = The dx calculations make a minor correction to increase accuracy
5	WaitDisable	This bit should always be set to '0.'
4	WriteBack bit	0 = ping-pong operation. For ping-pong operation, the line compositing operation must be completed in one video line time. When a given line of video is being read from one scanline buffer and sent to the display, the next line is being composed into the second scanline buffer.
		1 = write-back operation. Write-back operation is an alternate display mode to ping-pong operation. SOLO1 composites into one line buffer, and when that line is complete, the line is written back into a full frame buffer stored in SGRAM memory. The "spot mode" video DMA is then responsible for displaying the frame buffer on the screen. In this mode, the compositing operation is no longer bounded by the video line time.
3	FTB	Must always be programmed as a '1' for proper write-back operation.
21	RHSERVEID :	Reserved for fultine use
0	Soft Reset	0 = Soft Reset 1 = Force the gfxUnit and all of its state machines to reset. Must be cleared for unit to operate.

5.2.11.2 GRFX_OOTYCOUNT Register

This register stores the line count value from the yCounter.

Table 5-135 GRFX_OOTYCount Bit Definition (0x0400_6010)

Bits	Symbol	Meaning
31:10	RESERVED.	Reserved for future use.
9:0	OOTYCount	Ten-bit register that stores the value of the yCounter (line count) whenever a line runs out of time. To get the line number of the offending line, subtract 2 from this value. If it takes longer than a line time to service the GRFX_interrupt, it is possible that another line could have run out of time in the meantime. In this case, you will get the value for the last out-of-time line. (Read only)

5.2.11.3 GRFX_CELSBASE Register

This register provides a pointer to the first CelRecord in SGRAM.

Table 5-136 GRFX_CelsBase Bit Definition (0x0400_6014)

Bits	Symbol	Meaning
31:26	RESERVED	Reserved for future use:
25:2	CelsBase	Points to the first CelRecord in SGRAM. This register can be written by a special microCelRecord, as well as through normal R/W operation. The 2 LSB's are ignored and assumed to be zero.
1:0	RESERVED	Reserved for future use

5.2.11.4 GRFX_YMAPBASE Register

This register provides a pointer to the YMap structure in SGRAM.

Table 5-137 GRFX_YMapBase Bit Definition (0x0400_6018)

Bits	Symbol	Meaning
31-26	RESIDENTED TO	Reserved for future use
25:2	YMapBase	Points to the YMap structure in SGRAM. This register can be written by a special microCelRecord, as well as through normal R/W operation. The 2 LSB's are ignored and assumed to be zero.
1:0	RESERVED	Reserved for huture use

5.2.11.4.1 GRFX_CELSBASEMASTER

This register contains the CelBaseMaster value.

Table 5-138 GRFX_CelsBaseMaster Bit Definition (0x0400_601c)

Bits	Symbol	Meaning
31:26	RESERVE	Reserved for future use
25:2	CelsBaseMaster	At every HSYNC, the value in CelsBaseMaster is copied to CelsBase. This register can be written by a special microCelRecord, as well as through a normal write operation. The two LSB's are ignored and assumed to be zero.
1:0	RESERVED	Reserved for future use

5.2.11.5 GRFX_YMAPBASEMASTER

This register contains the value of the yMapBaseMaster.

Table 5-139 GRFX_YMapBaseMaster Bit Definition (0x0400_6020)

Bits	Symbol	Meaning
31:26	RESERVEDERE	Reserved for future use:
25:2	YMapBaseMaster	At every HSYNC, the value in yMapBaseMaster is copied to YMapBase. This register can be written by a special microCelRecord, as well as through normal R/W operation. The 2 LSB's are ignored and assumed to be zero.
10	RESERVED	Reserved for future use:

5.2.11.6 GRFX_INITCOLOR Register

This register contains the Scanline buffer's default YCrCb color value.

Table 5-140 GRFX_InitColor Bit Definition (0x0400_6024)

Bits	Symbol	Meaning
31:24	RESERVED	Reserved for future use:
23:0	InitColor	24 bit register which specifies the default YCbCr color of the scanline buffer. [23:16] Y value [15:8] Cb value [7:0] Cr value This register should only be modified during Vertical Blanking. Not to be confused with the VID_BLNK_COL register which specifies the color to be displayed during HBlank and VBlank.

5.2.11.7 GRFX_YCOUNTERINIT Register

This register is used to align the gfxUnit line counter with the VSYNC signal.

Table 5-141 GRFX_YCounterInit Bit Definition (0x0400_6028)

Bits	Symbol	Meaning
31:10	RESERVED	Reserved for future use
9:0	yCounterInit	Ten- bit signed number that specifies the value that the video line counter is initialized to at VSYNC (bit 10 is the sign bit). Program this register as follows: NTSC ping-pong = -274 (2EE) NTSC write-back = -420/2 (32E) PAL ping-pong = -332 (2B4) PAL write-back = -480/2 (310) The LSB of this register is ignored and replaced with '1' if the field is odd and '0' if the field is even. In this way, the internal yCounter counts frame lines instead of field line.

5.2.11.8 GRFX_PAUSECYCLES Register

This register provides an optional feature that can be used when in ping-pong mode. By programming this register, it is possible to force the gfxUnit to pause at the beginning of every horizontal line. This allows the CPU to use the memory bus during the pause without being interrupted by the gfxUnit.

Table 5-142 GRFX_PauseCycles Bit Definition (0x0400_602c)

Bits	Symbol	Meaning
31512	RESERVED	Reserved for future use
11:0	PauseCycles	12-bit register that specifies the number of cycles at the beginning of each line when SOLO1 will get off the bus.

5.2.11.9 GRFX_OOTCELSBASE Register

This register holds the value of the CelsBase register at the time of the last OOT interrupt.

Table 5-143 GRFX_OOTCelsBase Bit Definition (0x0400_6030)

Bits	Symbol	Meaning
31-24	KESEKVED	Reserved for future use
23:0	OOTCelsBase	24-bit register that stores the value of CelsBase whenever a line runs out of time. If it takes longer than a line time to service the GRFX_interrupt, it is possible that another line could have run out of time in the meantime. In this case, you will get the value for the last outOfTime line. (Read only)

5.2.11.10 GRFX_OOTYMAPBASE Register

This register holds the value of the yMapBase register at the time of the last OOT interrupt.

Table 5-144 GRFX_OOTYMapBase Bit Definition (0x0400_6034)

Bits	Symbol	Meaning
31.24	RESERVED	Reserved for future use
23:0	OOTYMapBase	24-bit register that stores the value of yMapBase whenever a line runs out of time. If it takes longer than a line time to service the GRFX_interrupt, it is possible that another line could have run out of time in the meantime. In this case, you will get the value for the last outOfTime line. (Read only)

5.2.11.11 GRFX_OOTCELSOFFSET Register

This register holds the value of the CelsOffset register at the time of the last OOT interrupt.

Table 5-145 GRFX_OOTCelsOffset Bit Definition (0x0400_6038)

Bit	Symbol	Meaning
31:11	RESERVED	Reserved for future use.
10:0	OOTCelsOffset	11-bit register that stores the value of CelsOff-set (where CelsBase + CelsOffset = address) whenever a line runs out of time. If it takes longer than a line time to service the GRFX_interrupt, it is possible that another line could have run out of time in the meantime. In this case, you will get the value for the last outOfTime line. (Read only)

5.2.11.12 GRFX_OOTYMAPCOUNT Register

This register holds the value of the yMapCount register at the time of the last OOT interrupt.

Table 5-146 GRFX_OOTYMapCount Bit Definition (0x0400_603c)

Bits	Symbol	Meaning
31:10	RESERVED.	Reserved for future use:
9:0	OOTYMapCount	10-bit register that stores the value of the yMap-Count (where yMapBase + yMapCount = address) whenever a line runs out of time. If it takes longer than a line time to service the GRFX_interrupt, it is possible that another line could have run out of time in the meantime. In this case, you will get the value for the last outOfTime line. (Read only)

5.2.11.13 GRFX_TERMCYCLECOUNT Register

This register is used only to evaluate performance It gives a measure of the time it took to compose the last line.

Table 5-147 GRFX_TermCycleCount Bit Definition (0x0400_6040)

Bits	Symbol	Meaning
31:12	RESERVED	Reserved for future use.
11:0	termCycleCount	12-bit register that stores the number of SYS2XCLK cycles from HSYNC to the processing of a termination yMap entry.

5.2.11.14 GRFX_HCOUNTERINIT Register

This register stores a value that represents one half of the number of pixels on an active line.

Table 5-148 GRFX_HCounterInit Bit Definition (0x0400_6044)

Bits	Symbol	Meaning
91410	RESERVED	Reserved for future use
9:0	HCounterInit	10-bit register that specifies 1/2 the number of pixels on an active line. For example, if there are 560 pixels on a line, the value would be 0x118 (dec 280).

5.2.11.15 GRFX_BLANKLINES and GRFX_ACTIVELINES Registers

The two registers, blankLines and activeLines, are used in "GRFX Direct Mode" to kick SOLO1 off of the bus during non-active video. Otherwise, the gfxUnit will tie up bus bandwidth by composing all of the lines during video blanking. Since these lines will never be displayed, it is no use composing them. Therefore, set BLANKLINES and ACTIVELINES to kick the gfxUnit off of the bus during vertical blanking.

In GRFX Direct Mode, the graphics engine only composes a line if the line is greater-than or equal-to BLANKLINES and less than ACTIVELINES + BLANKLINES.

Defaults are BLANKLINES = 0 and ACTIVELINES = 263. These defaults were set up for the GRFX Direct Mode. This means that SOLO1 is always accessing the bus for NTSC, even during vertical blanking.

For GRFX Write-back Mode, the ACTIVELINES register is used to tell the state machine how many lines there are in a *frame*. Remember that there are twice as many lines in a frame as there are in a field. Unfortunately, when the software changes between GRFX Write-back Mode and GRFX Direct Mode, it must change the value in the ACTIVELINES register.

Table 5-149 GRFX_BlankLines Bit Definition (0x0400_6048)

Bits	Symbol	Meaning
31:9 🛬	RESERVED	Reserved for future use:
8:0	blankLines	set blankLines equal to = (the number of lines from vertical sync to the end of vertical blanking - 2).

Table 5-150 GRFX_ActiveLines Bit Definition (0x0400_604c)

Bits	Symbol	Meaning
31.9	RESERVED	Reserved for future use.
8:0	activeLines	For GRFX Direct Mode: The number of active video lines in a field. For GRFX WriteBack Mode: The number of active video lines in a frame.

5.2.11.16 GFX_INTEN Register

This pair of registers enables the gfxUnit interrupts generated within SOLO1. The GFX_INTEN register sets the interrupt(s) and the GFX_INTEN_C register clears them.

Table 5-151 GFX_INTEN Bit Definitions (0x0400_6060, 0x0400_6064)

Bits	Symbol	Meaning
315	RESERVED A LOCAL	Reserved for future use
4	RANGE_INT_WBEOFL	Writeback has finished field
3	RANGE_INT_OOT	gfxUnit ran out of time compositing line
2	RANGE_INT_WBEOF	Writeback has finished frame
1:0	RESERVED	Reserved for titure use

5.2.11.17 GFX_INTSTAT Register

This pair of registers holds the status of all gfxUnit-generated interrupts. The GFX_INTSTAT register contains the interrupt status, and the GFX_INTSTAT_C register clears it.

Table 5-152 GFX_INSTAT Bit Definitions (0x0400_6068, 0x0400_606C)

Bits	Symbol	Meaning
31:5	RESERVED	Reserved for future sise
4	RANGE_INT_WBEOFL	Writeback has finished field
3	RANGE_INT_OOT	gfxUnit ran out of time compositing line
2	RANGE_INT_WBEOF	Writeback has finished frame
1:0	RESERVED	Reserved for future use

5.2.11.18 General Operation of the Write-back Unit

The first step in the operation of the Write-back Unit is to set up the GRFX structures in memory for normal GRFX mode operation. These are:

- celrecords,
- textures,
- ymap entries.

The other graphics registers, which are set up as usual, are:

- YCOUNTERINIT contains the top screen line to be processed.
 YMAPBASEMASTER, CELSBASEMASTER, and INITCOLOR contain proper values for the GRFX structures in memory.
- HCOUNTERINIT, BLANKLINES, and ACTIVELINES contain the screen size values. Note that for Write-back mode, the ACTIVELINES register is the number of frame lines, instead of the number of field lines. Therefore, it will be twice the value it would have been for non-Write-back mode.

The Write-back registers are set up as follows:

- The WBDCONFIG register is set to zero to start with, in order to shut down the unit and avoid unexpected behavior.
- The WBDSTART register is set to the start of the Write-back frame buffer in memory.
- The WBDLSIZE register is set to the number of bytes in a horizontal line.
- The Write-back bit is set in the GRFX_CONTROL register, which allows for Write back functionality

Other registers:

- The MEM_BURP register (0x0400_5008) must have the MEMBURPCT field set to a non-zero value. The larger the value, the better the gfxUnit performance, but with longer memory latency for the other units.
- Set potUnit to use gfxUnit (the UseGfx bit in the POT_CNTL register).

At this point, the gfxUnit can be started. The SCREENENABLE bit is set in the WBDCONFIG register, allowing for the generation of one complete frame (two fields). At this point, the Write-back unit is in operation. A flow-chart, illustrating the Write-back unit operation, is shown in Figure 5-7.

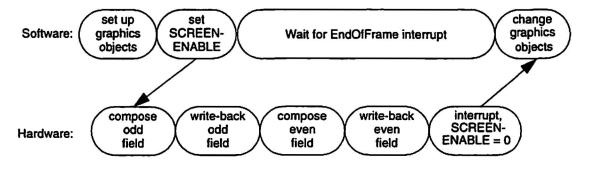


Figure 5-7 Write-back Unit Operation

At the completion of each frame, the EndOfFrame interrupt bit will be set in the GRFX_INTSTAT register, if the proper interrupts are enabled. When a bit in the INTSTAT register is set, the busUnit will be notified. The frame buffer will be complete at this time.

If you want the Write-back unit to continually run one frame after another, set the REPEATENABLE bit along with the SCREENENABLE bit. End continuous processing by clearing the two bits. The Write-back unit will go idle at the completion of the current frame.

To minimize problems interfacing to the video DMA unit, the Write-back unit scans the frame in an interlaced manor, i.e. all the odd lines are written in interleaved fashion to the frame buffer, followed by the even lines (see Figure 5-8). Currently, there is no provision for writing the buffer in a progressive manner.

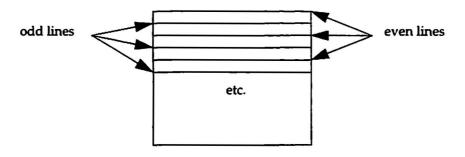


Figure 5-8 Interlaced Frame Scanning

To update the frame buffer field by field, set up the unit in the same manner as for frame operation, but use the ODDENABLE or EVENENABLE bits, rather than the SCREENENABLE bit. This results in only the odd or even lines, respectively, being generated and written back. At the completion of the field, the ENDOFFIELD interrupt bit will be set, if enabled.

Note: For SOLO1 Rev. 1.x, the starting location of the writeback frame buffer first field line must be set by software before each field rendering. Before issuing ODDENABLE, set the start to the top of the frame buffer, or before EVENENABLE, to the beginning of the second line in the frame buffer.

5.2.11.19 GFX_WBDSTART Register

The WBDStart register is only used in Write-back mode to specify where in memory the frame buffer data will be written by the Write-back unit. Normally, this will be the same value as the vidUnit register (VID_NSTART).

For SOLO1 Rev. 1.x, when writing back just the even field, the value would generally be (VID_NSTART + HSIZE*2), to point at the start of the even field in the frame buffer. The write-back unit will write back to the frame buffer at this location and the vidUnit will read out the frame buffer from this same location. Normally, this will be the same value as the vidUnit register VID_NSTART.

Table 5-153 WBDSTART Bit Definitions (0x0400_6080)

Bits	Symbol	Meaning
3126	RESERVED	Reserved for futtine use
25:2	WBDStart	This is the starting address of the writeback frame buffer in memory. It must be word aligned.
190	RESERVED.	The 2 LSB's are ignored and assumed to be zero

5.2.11.20 GFX_WBDLSIZE Register

The WBDLSize register is used in write-back mode only to set the width of the frame buffer that is written back to memory from the gfxUnit. Note that the width of the frame buffer must be an even value. The number of lines of the frame buffer which are written back to memory is set by the ACTIVELINES register. See the ACTIVELINES register definition.

Table 5-154 WBDLSize[9:0] Bit Definitions (0x0400_6084)

Bits	Symbol	Meaning
31:10	RESERVED	Reserved for future use.
9:0	WBDLSize	This is the number of pixels in one scan line of the frame buffer.

5.2.11.21 GFX_WBSTRIDE Register

This register is used in write-back mode only to set the offset from the end of one line to the start of the next line in the frame buffer. The offset is in pixels, and must be even.

Table 5-155 WBSTRIDE Bit Definitions (0x0400_608C)

Bits	Symbol	Meaning
31:10	RESERVED	Reserved for future use
9:0	WBSTRIDE	Offset value

5.2.11.22 GFX_WBDCONFIG Register

This register is used to configure the write-back unit.

Table 5-156 WBDCONFIG Bit Definitions (0x0400_6090)

Bits	Symbol	Meaning
314	RESERVED	Reserved for future use
3	Repeat Enable	Setting this bit results in the continuous operation of the write-back unit—the Enable bits are not cleared when the operation begins. Its value is not considered again until the completion of the current frame, and so cannot be used to stop an operation in progress.

Table 5-156 WBDCONFIG Bit Definitions (0x0400_6090)

Bits	Symbol	Meaning
2	ScreenEnable	When set to '1,' this bit tells the writeback unit to process one full screen (frame), i.e. two fields. As soon as the write-back operation begins, the bit is cleared if the Repeat Enable bit is not set. Its value is not considered again until the completion of the current frame, and so cannot be used to stop an operation in progress.
1	EvenEnable	When set to '1,' this bit instructs the write-back unit to process one even field. As soon as the write-back operation begins, the bit is cleared if the Repeat-Enable bit is not set. Its value is not considered again until the completion of the current frame, and so cannot be used to stop an operation in progress.
0	OddEnable	When set to '1,' this bit instructs the write-back unit to process one odd field. As soon as the write-back operation begins, the bit is cleared if the Repeat-Enable bit is not set. Its value is not considered again until the completion of the current frame, and so cannot be used to stop an operation in progress.

Note: SOLO1 Rev.2.x will have a software reset bit for the write-back unit.

5.2.12 divUnit Registers

Each "DIV_NEXT" register has a corresponding "DIV_CURR" register, that refers to the operation currently pending or being executed. These registers are for debugging purposes only, and should be considered read-only by software. All programming should be through the DIV_NEXT registers.

5.2.12.1 DIV_SYNCCNTL Register

This register controls the horizontal and vertical syncs.

Table 5-157 DIV_SYNCCNTL Bit Definitions (0x0400_8000)

Bits	Symbol	Meaning
31:22	RESERVED :	Reserved for future use.
21:20	syncSelect	2 = Extract sync information from the byte stream 1 = Extract sync information from embedded CCIR601 data 0 = Use explicit pins for sync
19	invertHS	1 = Invert sense of selected HS source
18	invertVS	1 = Invert sense of selected VS source
17	vsIsField	1 = Use the VS input as a field indicator
16	correctSyncs	1 = Enable forward error correction in the embedded syncs
1511	RESERVED	Reserved for future use
10:0	hsDelay	Number of clocks that recognition of the HS event is delayed

5.2.12.2 DIV_DMACNTL Register

This register controls the DMA operation.

Table 5-158 DIV_DMACNTL Bit Definitions (0x0400_8004)

Bits	Symbol	Meaning
31:17	RESERVED	Reserved for hiture use:
16	divReset	Reset the divUnit
15:4	RESERVED	Reserved for future use
3	pingPong	1 = Reload the DIV_NEXT* registers with the contents of DIV_CURR* when a DMA operation completes.
2	dmaEnabled	1 = Enable DMA operation

Table 5-158 DIV_DMACNTL Bit Definitions (0x0400_8004) (Continued)

Bits	Symbol	Meaning
1	nextValid	I = Indicates that the DIV_NEXT* registers have been loaded with a correctly described DMA operation.
0	nextValidForever	1 = Always load DIV_CURR* registers with DIV_NEXT*, i.e. keep repeating a single DMA operation.

5.2.12.3 DIV_NEXTVBITB Register

This register defines the starting and ending line numbers for the VBI DMA operation.

Table 5-159 DIV_NEXTVBITB Bit Definitions (0x0400_8008)

Bits	Symbol	Meaning
3126	RESERVED (Reserved for future use
25:16	nextVBIStartline	First valid VBI line
15:10	RESERVED	Reserved for future use
9:0	nextVBIEndLine	Last valid VBI line

5.2.12.4 DIV_NEXTVBILR Register

This register defines the number of clocks that occur, following horizontal sync, before the first valid pixel appears at the data output.

Table 5-160 DIV_NEXTVBILR Bit Definitions (0x0400_800C)

Bits	Symbol	Meaning
3197	RESERVED E	Reserved for future use:
26:16	nextVBIBack- Porch	Number of clock ticks from the HS event before the first valid pixel appears at the data input.
159	RESHRVED	Reserved/for name uses
8:0	nextVBIWidth	Number of pixel-pairs of active VBI to extract from each line. Width = #pixelsPerLine/2.

5.2.12.5 DIV_NEXTVBIADDR Register

This register defines the starting memory address where VBI data will be written.

Table 5-161 DIV_NEXTVBIADDR Bit Definitions (0x0400_8010)

Bits	Symbol	Meaning
31:26	RESERVED	Reserved for future use.
25:5	nextVBIAddr	Starting VBI memory address
4:0	RESERVED	Reserved for future use

5.2.12.6 DIV_NEXTTB Register

This register defines the starting and ending active video lines.

Table 5-162 DIV_NEXTTB Bit Definition (0x0400_8014)

Bits	Symbol	Meaning
31:26	RESERVED	Reserved for future use.
25:16	nextVideoStartline	First valid video line
15:10	RESERVED	Reserved for future use.
9:0	nextVideoEndLine	Last valid video line

5.2.12.7 DIV_NEXTLR Register

This register defines the number of clocks that occur, following horizontal sync, before the first valid pixel appears at the data output.

Table 5-163 DIV_NEXTLR Bit Definitions (0x0400_8018)

Bits	Symbol	Meaning
31:27	RESERVED #	Reserved for future use
26:16	nextBackPorch	Number of clock ticks from the HS event before the first valid pixel appears at the data input.
15:9	RESERVED	Reserved for future use
8:0	nextWidth	Number of pixel-pairs of active video to extract from each line. Width = #pixelsPerLine / 2.

5.2.12.8 DIV_NEXTCFG Register

This register defines which field will be captured by the current DMA operation.

Table 5-164 DIV_NEXTCFG Bit Definitions (0x0400_801C)

Bits	Symbol	Meaning
31:30	RESERVED	Reserved for future use.
29	nextVBIEnabled	Next VBI field is enabled
28	nextVideoEnabled	Next video field is enabled
27:25	RESERVED	Reserved for future use
24	nextField	Indicates which field this DMA operation is to capture.
23	RESERVED	Reserved for future use
22:20	nextRsmpMode	Resampling mode for this operation: 000 = none 010 = 9 to 8 (720 -> 640) 011 = 15 to 16 (720 -> 768) 100 = 2 to 1 (720-> 360) 110 = 9 to 4 (720 -> 320) 111 = 5 to 8 (720 -> 384)
19:17	RESERVED	Reserved for future use
16	nextDecEn	Enable the 2-to-1 decimator for this operation (this is in addition to the resampling mode).
15:5	nextStride	Number of 32-byte chunks from the starting address of each active video line to the next.
4:0	RESERVED	Reserved for future use

5.2.12.9 DIV_NEXTADDR Register

This register supplies the starting address for the DMA operation being described.

Table 5-165 DIV_NEXTADDR Bit Definitions (0x0400_8020)

Bits	Symbol	Meaning
31-26	RESERVIED	Reserved for hiture use
25:5	nextAddr	Starting address for the video DMA operation being described.
40 = 5	RESERVED	Reserved for future use.

5.2.12.10 DIV_CURRVBITB Register

This register defines the starting and ending line numbers for the VBI DMA operation.

Table 5-166 DIV_CURRVBITB Bit Definitions (0x0400_8024)

Bits	Symbol	Meaning
31:26,	RESERVED	Reserved for future use:
25:16	vbiStartLine	First valid VBI line.
15:10	RESERVED	Reserved for future use
9:0	vbiEndLine	Last valid VBI line.

5.2.12.11 DIV_CURRVBILR Register

This register defines the number of clocks that occur, following horizontal sync, before the first valid pixel appears at the data output.

Table 5-167 DIV_CURRVBILR Bit Definitions (0x0400_8028)

Bits	Symbol	Meaning
31-27	RESERVED	Reserved for future use
26:16	VBIBackPorch	Number of clock ticks from the HS event before the first valid pixel appears at the data input.
15:9	RESERVED	Reserved for inture use
8:0	nextWidth	Number of pixel-pairs of active VBI to extract from each line. Width = #pixelsPerLine/2.

5.2.12.12 DIV_CURRVBIADDR Register

This register defines the current starting memory address where VBI data will be written.

Table 5-168 DIV_CURRVBIADDR Bit Definitions (0x0400_802C)

Bits	Symbol	Meaning
3126	RESERVED	Reserved for future use
25:5	vbiAddr	Starting memory address
40.	RESERVED	Reserved for future use

5.2.12.13 DIV_CURRTB Register

This register defines the starting and ending active video lines.

Table 5-169 DIV_CURRTB Bit Definitions (0x0400_8030)

Bits	Symbol	Meaning
31:26	RESERVED:	Reserved for future use:
25:16	videoStartLine	Current first valid video line
15:10	RESERVED	Reserved for future use
9:0	videoEndLine	Current last valid video line

5.2.12.14 DIV_CURRLR Register

This register defines the number of clocks that occur, following horizontal sync, before the first valid pixel appears at the data output.

Table 5-170 DIV_CURRLR Bit Definitions (0x0400_8034)

Bits	Symbol	Meaning
3126	RESERVED	Keserved for future use
25:16	backPorch	Number of clock ticks from the HS event before the first valid pixel appears at the data input.
15:10	RESERVED	Reserved for niture use.
9:0	width	Number of pixel-pairs of active video to extract from each line. Width = #pixelsPerLine/2.

5.2.12.15 DIV_CURRCFG Register

This register defines which field will be captured by the current DMA operation.

Table 5-171 DIV_CURRCFG Bit Definitions (0x0400_8038)

Bits	Symbol	Meaning
31:30	RESERVED 6	Reserved for future use
29	currVBIEnabled	Next VBI field is enabled
28	currVideoEnabled	Next video field is enabled
27:25	RESERVED 1	Reserved for future use ≥
24	currField	Indicates which field this DMA operation is to capture.
23	TRESERVED	Reserved for future use

Table 5-171 DIV_CURRCFG Bit Definitions (0x0400_8038) (Continued)

Bits	Symbol	Meaning
22:20	currRsmpMode	Resampling mode for this operation: 000 = none 010 = 9 to 8 (720 -> 640) 011 = 15 to 16 (720 -> 768) 100 = 2 to 1 (720->360) 110 = 9 to 4 (720 -> 320) 111 = 5 to 8 (720 -> 384)
19:17	RESERVED	Reserved for future use:
16	currDecEn	Enable the 2-to-1 decimator for this operation (this is in addition to the resampling mode).
15:5	currStride	Number of 32-byte chunks from the starting address of each active video line to the next.
4:0	RESERVED	Reserved for future use:

5.2.12.16 DIV_CURRADDR Register

This register supplies the starting address for the DMA operation being described.

Table 5-172 DIV_CURRADDR Bit Definitions (0x0400_803C)

Bits	Symbol	Meaning
31:26	RESHRVED	Reserved for future use
25:5	currAddr	Starting address for the video DMA operation being described.
40	RESERVED	Reserved for future use

5.2.12.17 DIV_AUDCNTL Register

This register configures the timing of the audio signals.

Table 5-173 DIV_AUDCNTL Bit Definitions (0x0400_8040)

Bits	Symbol	Meaning	
31	bclkEdgeSelect	Selects the edge to capture audio data. 0=Rising 1=Falling	
30	audFlipBits	Selects order of incoming data. 0=MSB first 1=LSB first	

Table 5-173 DIV_AUDCNTL Bit Definitions (0x0400_8040) (Continued)

Bits	Symbol	Meaning	
29	audInvertMSB	Determines arithmetic format for data. 0=Two's complement 1=Offset binary	
28	audInvertLR- Clk	Inverts the LRCLK input. 0=Invert 1=Don't invert	
27	audSwapLR	Swaps the left and right channels. 0=No swap 1=Swap	
26	audCaptu- reEnable	Enables audio capturing. 0=No enable 1=Enable	
25:21	RESERVED	Reserved for future use	
20:16	audWord- Width	Number of bits per sample -2. Typical value =16.	
15:4	RESERVED	Reserved for future use	
3	audPingPong	Reloads the DIV_NEXTAUD* registers with the contents of the DIV_CURRAUD* registers when a DMA operation is performed. 0=No reload 1=Reload	
2	audEnabled	Enables Divit audio. 0=No enable 1=Enable	
1	audNextValid	Reloads the DIV_NEXTAUD* registers with valid data. 0=No reload 1=Reload	
0	audNextValid- Forever	Always loads the DIV_CURRAUD* registers with the contents of the DIV_NEXTAUD* registers (i.e., keeps repeating the operation). If used with aud-PingPong, the DMA will continuously cycle between two buffers. 0=No load 1=Load	

5.2.12.18 DIV_NEXTAUDADDR Register

This register contains the location of the next audio start address.

Table 5-174 DIV_NEXTAUDADDR Bit Definitions (0x0400_8044)

Bits	Symbol	Meaning
31:26	RESERVED	Reserved for future use
25:5	nextAudAddr	Next audio start address.
4:0	RESERVED	Reserved for future use

5.2.12.19 DIV_NEXTAUDLEN Register

This register contains the number of audio samples to read in.

Table 5-175 DIV_NEXTAUDLEN Bit Definitions (0x0400_8048)

Bits	Symbol	Meaning
31:21	RESERVED	Reserved for future use
22:5	nextAudLen	Next audio length in samples.
40	RESERVED	Reserved for future use

5.2.12.20 DIV_CURRAUDADDR Register

This register contains the location of the current audio start address.

Table 5-176 DIV_CURRAUDADDR Bit Definitions (0x0400_804C)

Bits	Symbol	Meaning
31:26	RESERVED.	Reserved for future use
25:5	audAddr	Current audio start address.
40	RESERVED	Reserved for future use.

5.2.12.21 DIV_CURRAUDLEN Register

This register contains the number of audio samples to read in.

Table 5-177 DIV_CURRAUDLEN Bit Definitions (0x0400_8050)

Bits	Symbol	Meaning
31:21	RESERVED	Reserved for future use
22:5	audLen	Current audio length in samples.
4:0	RESERVED	Reserved for future use

5.2.12.22 DIV_SYNCPHASE Register

This register is a counter used by software to keep the output video synchronized with the input video. It starts counting at the beginning of DIVIT's odd field, and stops when the potUnit begins its odd field—measuring the phase difference between the input and output video.

Table 5-178 DIV_SYNCPHASE Bit Definitions (0x0400_8060)

Bits	Symbol	Meaning
31:24	RESERVED	Reserved for future use
23:0	syncPhase	The value of the phase difference between input and output video, measured in system clocks.

5.2.12.23 DIV_GPOEN Register

This register resets all of the GPIO inputs. Each bit can be programmed as an output.

Table 5-179 DIV_GPOEN Bit Definitions (0x0400_8064)

Bits	Symbol	Meaning
31:25	RESERVED:	Reserved for future use
24	div_pwmEnable	Enables the div_pwm signal to drive dev_GPIO[8].
23:16	pwmLevel	Controls the width of the pwm clock.
15:13	RESERVED	Reserved for future use
12:5	DIV_DATA_en	Enables these bits for GPIO output.
4	DIV_HS_en	Enables this bit for GPIO output.
3	DIV_VS_en	Enables this bit for GPIO output.
2	DIV_BCLK_en	Enables this bit for GPIO output.
1	DIV_LRCLK_en	Enables this bit for GPIO output.
0	DIV_SDATA_en	Enables this bit for GPIO output.

5.2.12.24 DIV_GPO Register

This register contains the value of the output bits, if configured as such by the DIV_GPOEN register.

Table 5-180 DIV_GPO Bit Definitions (0x0400_8068)

Bits	Symbol	Meaning
31:13	RESERVED	Reserved for future use.
12:5	DIV_DATA	Specifies a value for the GPIO output (if enabled).
4	DIV_HS	Specifies a value for the GPIO output (if enabled).
3	DIV_VS	Specifies a value for the GPIO output (if enabled).
2	DIV_BCLK	Specifies a value for the GPIO output (if enabled).
1	DIV_LRCLK	Specifies a value for the GPIO output (if enabled).
0	DIV_SDATA	Specifies a value for the GPIO output (if enabled).

5.2.12.25 DIV_GPI Register

This register contains the state of each input, regardless of its configuration.

Table 5-181 DIV_GPI Bit Definitions (0x0400_806c)

Bits	Symbol	Meaning
31:13	RESHRVED.	Reserved for fultire use.
12:5	io_DIV_DATA	GPIO input value.
4	io_DIV_HS	GPIO input value.
3	io_DIV_VS	GPIO input value.
2	io_DIV_BCLK	GPIO input value.
1	io_DIV_LRCLK	GPIO input value.
0	io_DIV_SDATA	GPIO input value.

5.2.13 dveUnit Registers

This section provides descriptions for each of the dveUnit registers. For more information on the internal workings of the dveUnit, please refer to the "DVE 2.0 Theory of Operation."

5.2.13.1 DVE_CNTL Register

This register controls the operation of the dveUnit.

Table 5-182 DVE_CNTL Bit Definitions (0x0400_7000)

Bits	Symbol	Meaning
314	RESERVED	Reserved for future use
3	QualInvert	Invert the Clock Qualification signal (used for debug only).
2	TGEN	Timing Generation Enable. The dveUnit will create HSYNC and VSYNC. When set, be sure that the V/HSYNC output enables are set in the potUnit.
1	VEN	dveUnit Enable. This is also connected to the reset line of the DVE engine.
0	CEN	Clock Divider Enable. When using the dveUnit, POT_CLK is a 4x clock. Set this bit to '1' to provide a 2x clock to the rest of the chip.

5.2.13.2 DVE_CNFG Register

This register contains the dveUnit configuration bits.

Table 5-183 DVE_CNFG Bit Definitions (0x0400_7004)

Bits	Symbol	Meaning
31528	RESERVED	Reserved to suture use
22	SYNCBYPASS	Removes the syncs from the video output signals
21	DACPDN	DAC power-down. When cleared, turns off the drive power to the DACS. Must be set for normal video operation.
20	DACORDER	When set, inverts the MSb-LSb ordering to the DACS. Must be cleared for SOLO1.
19	DACBYPASS	When set, allows DIVIT to drive the DACS directly. When cleared, DVE drives the DACS.
18	DTMEN	Special non-interlace mode. Should be set to zero.

Table 5-183 DVE_CNFG Bit Definitions (0x0400_7004) (Continued)

Bits	Symbol	Meaning
17	TPGEN	Generates standard color bars in RGB space, rather than using supplied system pixel data. MATBYPAS-SEN must be cleared for proper operation.
16	PORCH	Adjusts the front and back porch to meet CCIR-624-4 when in non-square pixel mode. Should be set to zero.
15	SETUPEN	Enables 7.5 IRE setup step for NTSC. No effect for PAL. Should be set to '1.'
14	MATBYPAS- SEN	Bypasses the RGB -> YUV input conversion matrix. Must be set for normal SOLO1 video.
13	DRIVECSYNC	Drives composite sync on the VSYNC output pin.
12	BLANK	Video blanking disable. Must be set for SOLO1 video output.
11	INVERTDAC	If set, inverts the values of all signals going to the DACS. Must be cleared for SOLO1.
10	COLORKILL	Blanks the burst and chroma signals at the DACs. Should be '0.'
9	DACDITHEN	Dither enable. Provides 10-bit output with 9-bit DACs. Must be '0' for SOLO1.
8	RGBDITHEN	RGB dither enable. Enables dithering for RGB output data. Must be cleared for YCbCr data.
7	FMODE	Filter mode. Should only be set for S-Video.
6	FIXEDIRE	If set, uses the fixed DAC output levels. If cleared, uses the entire range of the DACS.
5	BYPASSEN	DVE bypass. If set, input data is passed directly to outputs. Must be '0.'
4	SYNCLOCK	Sync Slave mode enable. Allows the DVE to slave to an externally supplied VSYNC/HSYNC. Should be '0.'
3	RGBout	Selects for RGB or composite output.
2	PAL	Selects PAL or NTSC output.
1	Progressive	Selects progressive or interlaced output. Should be set to '0.'
0	NONSQ	Non-square pixels. Must be cleared for SOLO1.

5.2.13.3 DVE_DBDATA Register

This register holds values to be loaded into the dveUnit counters for debugging and synchronization. Depending upon how the DVE_DBEN register is configured (i.e., which fields are enabled), either the 24-bit Phase Value, or the three separate Pixel, Line, and Field values can be loaded.

Table 5-184 DVE_DBDATA Bit Definitions (0x0400_7008)

Bits	Symbol	Meaning
31:24	RESERVED.	Reserved for future use
23:0	PhaseVal	Color burst phase value. Selected if the PhaseLd bit, in the DVE_DBEN register, is enabled.
23:21 20:11 10:0	FieldVal LineVal PixelVal	Load Field Value Load Line Value Load Pixel Value Selected if the corresponding bits, in the DVE_DBEN register, is enabled.

5.2.13.4 DVE_DBEN Register

This register controls the Phase, Field, Line and Pixel Value enables. When a bit in this register is set, the corresponding field in the DBDATA register is loaded into the appropriate internal DVE counter, and the bit is cleared. Any number of this register's bits may be set at once, although not all combinations make sense. Do to its pulse nature, this register is write-only.

Table 5-185 DVE_DBEN Bit Definitions (0x0400_700C)

Bits	Symbol	Meaning
314	RESERVED	Reserved for future use
3	FieldLd	Load Field Value
2	LineLd	Load Line Value
1	PixelLd	Load Pixel Value
0	PhaseLd	Load Phase Value

5.2.13.5 DVE_DTST Register

This register allows DAC testing by driving a supplied value to a chosen DAC.

Table 5-186 DVE_DTST Bit Definitions (0x0400_7010)

Bits	Symbol	Meaning
31:13	RESERVED	Reserved for future use:
12:3	DacData	Data to pass to the DAC.
2:1	DacSel	Select the DAC to receive the data.
0	DacTest	Enable the DAC Test mode.

5.2.13.6 DVE_RDFIELD Register

This register returns the current dveUnit counter values. It is read-only.

Table 5-187 DVE_RDFIELD Bit Definitions (0x0400_7014)

Bits	Symbol	Meaning
31:24	RESERVED	Reserved for future use
23:21	FieldVal	Current Field Value
20:11	LineVal	Current Line Value
10:0	PixelVal	Current Pixel Value

5.2.13.7 DVE_RDPHASE Register

This register returns the current color burst phase value. It is read-only.

Table 5-188 DVE_RDPHASE Bit Definitions (0x0400_7018)

Bits	Symbol	Meaning
3124	RESERVED	Reserved for future use
23:0	PhaseVal	Current Phase Value

5.2.13.8 DVE_FILTCNTL Register

This register controls the dveUnit output filter.

Table 5-189 DVE_FILTCNTL Bit Definitions (0x0400_701C)

Bits	Symbol	Meaning
SIJS.	RESERVED ::.	Reserved for future use
12	C0	Filter coefficient
11:9	C1	Filter coefficient

Table 5-189 DVE_FILTCNTL Bit Definitions (0x0400_701C) (Continued)

Bits	Symbol	Meaning
8:6	C2	Filter coefficient
5:3	C3	Filter coefficient
2:1	C4	Filter coefficient
0	Filter	Filter Enable

5.2.14 potUnit Registers

The potUnit (Pixel Output and Timing) drives both interlaced and non-interlaced NTSC or PAL displays. There are several registers that control the various features of the potUnit, and control the actual frame being displayed on the screen. Both the horizontal and vertical resolution of the video display are programmable. The potUnit is basically a timing engine that takes pixels from some source and displays them to the screen. A TV has a resolution of 'X' pixels by 'Y' lines (see Figure 5-9). For NTSC, X = 640 and Y = 480, but not all of these pixels are visible since they may be in the blanking time or the overscan region of the TV. SOLO1 allows software to position the active window anywhere on the overall X and Y coordinates of the TV, so that all pixels are guaranteed to be visible. This is done by programming the HSTART, HSIZE, VSTART, and VSIZE registers in SOLO1.

The potUnit can sink from two internal sources, the gfxUnit, and the vidUnit. In both cases, the potUnit supplies a pixel advance signal to the units.

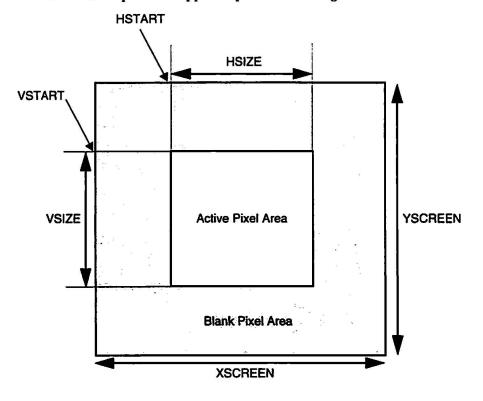


Figure 5-9 Video Display

5.2.14.1 POT_VSTART Register

This register is used to set the vertical starting position of active lines on the screen. The value in this register, plus the values in the HSIZE, HSTART, and VSIZE need to match the appropriate DMA buffer size that software must set up (hardware does not check that all values match). This is in frame lines, not field lines.

Table 5-190 POT_VSTART Bit Definitions (0x0400_9080)

Bits	Symbol	Meaning
31.11	RESERVED	Reserved for future use
10:0	VSTART[10:0]	This field reflects the vertical starting line out of a possible 525 (NTSC) or 625 (PAL).

5.2.14.2 POT_VSIZE Register

This register is used to set the number of lines on the screen. This is in frame lines, not field lines.

Table 5-191 POT_VSIZE Bit Definitions (0x0400_9084)

Bits	Symbol	Meaning
Side :	PRIESERAMED FOR	Reserved for finance use
10:0	VSIZE[10:0]	This field reflects the vertical size of the active area. The VSTART + VSIZE value must not be greater than 525 (NTSC) or 625 (PAL), or unpredictable results will occur. The "safe" vertical area is 420 for NTSC and 480 for PAL.

5.2.14.3 POT_BLNKCOL Register

This register sets the default color value to be displayed if this feature is enabled.

Table 5-192 POT_BLNKCOL Bit Definitions (0x0400_9088)

Bits	Symbol	Meaning
8124	RESTRANTED.	Reserved for future use
23:16	YCOL	Y value to be displayed for each pixel during blank times.
15:8	CRCOL	Cr value to be displayed for each pixel during blank times.
7:0	CBCOL	Cb value to be displayed for each pixel during blank times.

5.2.14.4 POT_HSTART Register

This register is used to set the horizontal starting position of active pixels on the screen. The value in this register, plus the values in the HSIZE, VSTART, and VSIZE registers, need to match the appropriate DMA buffer size that software must set up (hardware does not check that all values match). This is measured in pixels. For external encoders, this value must be even.

Table 5-193 POT_HSTART Bit Definitions (0x0400_908C)

Bits	Symbol	Meaning
31413°	RESERVED	Reserved for future use
10:0	HSTART[10:0]	This field reflects the horizontal starting pixel (out of a possible 640 (NTSC) or 768 (PAL)).

5.2.14.5 POT_HSIZE Register

This register is used to set the horizontal pixel size of the screen. It is measured in pixels.

Table 5-194 POT_HSIZE Bit Definitions (0x0400_9090)

Bits	Symbol	Meaning .
3131	RESERVED	Reserved for future use
10:0	HSIZE[10:0]	This field reflects the horizontal screen size of the active area. The HSTART + HSIZE value must not be greater than 640 (NTSC) or 768 (PAL), or unpredictable results will occur. The actual "safe" horizontal area is 560 for NTSC and 640 for PAL. Must be a multiple of 8 pixels (4 words).

5.2.14.6 POT_CNTL Register

This register controls the various potUnit functions.

Table 5-195 POT_ CNTL Bit Definitions (0x0400_9094)

Bits	Symbol	Meaning
erin ,	RESERVED	Reserved for future use
11	UseGfx444	When sourcing the dveUnit, use 4:4:4 data from the gfxUnit, rather than interpolation. This is the preferred operational mode for SOLO1.
10	DVECCS	Selects which edge of CrCbSel is used to latch data for GRFX->DVE interpolation. Since interpolation isn't used, the setting doesn't matter.

Table 5-195 POT_ CNTL Bit Definitions (0x0400_9094) (Continued)

Bits	Symbol	Meaning
9	DveHalfShift	Shifts the pipeline to the dveUnit 1/2 pixel. (For debug only)
8	HfieldLine	0 = Hint is in frame lines 1 = Hint is in 2x field lines
7	VidSoutEn	Enables the video sync output pins (resets to '0'). Note: when DVE_TEN is set, this bit must be set.
6	VidDoutEn	Enables the video output pins (resets to '1'). Note: When DVE_TEN is set, this bit must be set.
5	HalfShift	Shifts the external encoder pixel pipeline 1/2 pixel (used for debugging)
4	InvertCrCb	Inverts the MSB of CrCb (same as adding 128).
3	UseGfx	Sets up the gfxUnit as a source, rather than the vidUnit.
2	SoftReset	Enables a soft reset of the potUnit
1	Progressive	Sets up the potUnit for progressive (non-interlaced) mode.
0	EnableOutputs	Enables the potUnit.

5.2.14.7 POT_HINTLINE Register

This register is used to set the line that generates an interrupt, if the VID_HSYNCEN bit is set in the VID_INTEN register.

Table 5-196 POT_ HINTLINE Bit Definitions (0x0400_9098)

Bits	Symbol	Meaning
10:0	RESERVED HINT- LINE[10:0]	Reserved for future use. This field reflects the line that causes the interrupt if the horizontal interrupt is enabled. After reset, this field is 0x000. In Field Mode, Bit 0 is a 'don't-care.' If this field is set to '7' in Field Mode, the third dis-
		played line in each field will generate an interrupt. If this field is set to '7' in Frame Mode, the third displayed line in the odd field will generate an interrupt.

5.2.14.8 POT_INTEN Register

This register enables the potUnit interrupts.

Table 5-197 POT_ INTEN Bit Definitions (0x0400_909C)

Bits	Symbol	Meaning
31:6	RESERVED	Reserved for future use.
5	VIDVSYNCEEN	When '1,' this bit enables even-field vertical sync interrupts to pass through and be seen by the CPU. When '0,' this interrupt source is disabled. Software must write '1's to the POT_INTEN_S location (to set the bit), and to the POT_INTEN_C location (to clear the bit). This bit is set to '0' after reset.
4	VIDVSYNCOEN	When '1,' this bit enables odd-field vertical sync interrupts to pass through and be seen by the CPU. When '0,' this interrupt source is disabled. Software must write '1's to the POT_INTEN_S location (to set the bit), and to the POT_INTEN_C location (to clear the bit). This bit is set to '0' after reset.
3	VIDHSYNCEN	When '1,' this bit enables horizontal sync interrupts to pass through and be seen by the CPU. When '0,' this interrupt source is disabled. Software must write '1's to the POT_INTEN_S location (to set the bit), and to the POT_INTEN_C location (to clear the bit). This bit is set to '0' after reset.
2	Shift	Interrupt when shiftage occurs (i.e., no valid pixels from vidUnit when read).
10	CHANNED :	Reserved for future use

5.2.14.9 POT_INTSTAT Register

This register contains the status of the potUnit interrupts.

Table 5-198 POT_ INTSTAT Bit Definitions (0x0400_90A0)

Bits	Symbol	Meaning
31:6	RESERVED-	Reserved for future use.
5	VIDVSYNCE	When this bit is set to '1' a potUnit interrupt has occurred due to a vertical sync indicating the start of an even field. Clear the interrupt by writing a '1' to the corresponding bit in the POT_INTSTAT_C register.
4	VIDVSYNCO	When this bit is set to '1' a potUnit interrupt has occurred due to a vertical sync indicating the start of an odd field. Clear the interrupt by writing a '1' to the corresponding bit in the POT_INTSTAT_C register.
3	VIDHSYNC	When this bit is set to '1,' a potUnit interrupt has occurred due to a horizontal sync occurring on line 'n,' where 'n' is the line defined in the VID_HINTLINE register. It is the HSYNC after the HINTLINE is displayed that triggers this interrupt. Clear this interrupt by writing a '1' to the corresponding bit in the POT_INTSTAT_C register.
2	Shift	When this bit is set to '1' a potUnit interrupt has occurred due to a shift in the displayed screen, caused by a lack of valid data from the vidUnit when requested by the potUnit. Note that in SOLO1, this bit will always read back as zero, even though it is functioning correctly. If a potUnit interrupt occurs, and there are no bits set in this register, a Shift interrupt may be inferred.
2:0	म्यद्भावस्थाति	Reserved for furme use:

Note: For software testing, any of these bits may be set by writing a '1' to the corresponding bit in the POT_INTSTAT_S register.

5.2.14.10 POT_CLINE Register

This register is used to read the current line being displayed.

Table 5-199 POT_ CLINE Bit Definitions (0x0400_90AC)

Bits	Symbol	Meaning
31:11	RESERVED	Reserved for future use.
10:0	CLINE[10:0]	This field reflects the line currently being displayed. Note that CLINE[0] indicates whether or not the current line is in an odd field (1) or an even field (0).

5.2.14.11 POT_INTEN_C Register

This register clears the pot Unit interrupts.

Table 5-200 POT_INTEN_C Bit Definitions (0x0400_90A4)

Bits	Symbol	Meaning
31:6	RESERVED	Reserved for future use
5	VIDVSYNCEEN	When '1,' this bit clears the even-field vertical sync interrupts.
4	VIDVSYNCOEN	When '1,' this bit clears the odd-field vertical sync interrupts.
3	VIDHSYNCEN	When '1,' this bit clears the horizontal sync interrupts.
2	Shift	When '1,' this bit clears the shift interrupts.
1.0	RESERVED	Reserved for future use.

5.2.14.12 POT_INTSTAT_C Register

This register clears the status of the potUnit interrupts.

Table 5-201 POT_ INTSTAT_C Bit Definitions (0x0400_90a8)

Bits	Symbol	Meaning
31.6	RESERVED	Reserved for future use
5	VIDVSYNCE	Clears the interrupt status set by the corresponding bit in the POT_INTSTAT register.
4	VIDVSYNCO	Clears the interrupt status set by the corresponding bit in the POT_INTSTAT register.
3	VIDHSYNC	Clears the interrupt status set by the corresponding bit in the POT_INTSTAT register.

Table 5-201 POT_INTSTAT_C Bit Definitions (0x0400_90a8) (Continued)

Bits	Symbol	Meaning
2	Shift	Clears the interrupt status set by the corresponding bit in the POT_INTSTAT register.
1:0	RESERVED	Reserved for future use

5.2.14.13 Typical potUnit Programming

This section provides an example of what typical programming for the potUnit might look like:

This example assumes that there is already a source for pixels primed—either the vidUnit or the gfxUnit.

Once the potUnit is enabled, it will wait for an odd VSYNC. At that point, it will request the first pixel, and wait for the active video region. Then, it will request pixels until the end of the frame.

Once video has started, clearing the POTEN bit will have no effect until the end of the frame (i.e. video will stop at the end of the frame). If video needs to be stopped immediately, set the SOFTRESET bit, which will clear the internal state of the unit.

The system may wish to be notified at period times, such as the start of a certain field, the start of a certain field or frame line, or when slippage may have occurred. The general method for interrupt handling is shown in the following example code (this code is for an HLINE interrupt).

```
*(kBUS_VIDINTEN_S) = kV_BUS_VIDINT_POT;// enable pot interrupts
                                          to system
*(kPOT_CNTL) |= kV_POT_CNTL_HLINEFIELD; // hline value is for
                                          field, not frame
*(kPOT_HINTLINE) = kHLineToCatch;
                                       // which line we want to
                                          catch
*(kPOT_INTEN_S) = kV_POT_INT_HLINE;
                                       // enable the interrupt
(wait for interrupt)
*(kPOT_INTEN_C) = kV_POT_INT_HLINE;
                                       // clear the enable
*(kPOT_INTSTAT_C) = kV_POT_INT_HLINE; // clear the interrupt bit
*(kBUS_VIDINTSTAT_C) = kV_BUS_VIDINT_POT;
                                       // clear the potUnit
                                          interrupt to the
                                          system
```

Note: In version 1.0 of SOLO1, the SHIFT interrupt bit is not reflected in the status register. So, if that interrupt source is enabled, an interrupt occurs, and no bit in the potUnit status register is set, then a SHIFT interrupt can be assumed to have fired.

5.2.15 sucUnit Registers

The sucUnit has two identical register sets. One supports the Smart Card interface (SC0) and the other supports a general-purpose UART (GPU). The registers comprising each set are separated into the following eight categories:

- 1. UART FIFO Registers
- 2. Shift Register Registers
- 3. Clock Divider Registers
- 4. Line Control and Status Registers
- 5. Interrupt Registers
- 6. Smart Card Deactivation Registers
- 7. GPIO Registers
- 8. Diagnostic Registers

5.2.15.1 UART FIFO Registers

5.2.15.1.1 SUCGPU_TFFHR/ SUCSC0_TFFHR Registers

The Transmit FIFO Hold register set is used to access the UART transmit FIFO. In normal mode, the FIFO is writable by the CPU, but not readable. The FIFO data is read by the transmit shift register and serially shifted out on the UART_TXD/SC0_DATA signal.

Table 5-202 SUCGPU_TFFHR/SUCSCO_TFFHR Bit Definitions (0x0400_A000, 0x0400_A800)

Bits	Symbol	Meaning
31:9	RESERVED :-	Reserved for future use
8	tffError	Transmit FIFO Error. The tffError field is ignored for transmission. It is not read by the transmit shift register. If the FIFO is software visible, Bit 8 of the CPU write data will be written to the FIFO unless the write causes a FIFO overflow. It is forced to '1' in the case of a FIFO overflow. State after reset is 0x00. A software FIFO reset, writing a '1' to ffReset (TFFCR[0]) will set all tffError bits in the FIFO to '0.'
7:0	tffData	Transmit FIFO Data. This field contains the data that will be read by the transmit shift register to shift out serially. All entries are set to 0x00 on reset. The state is not affected by a software FIFO reset writing a '1' to ffReset (TFFCR[0]).

When ffSWVisible (TFFCR[2]) is set to one, the FIFO is readable and writable from the CPU. 'Write' operations will write to the tail of the FIFO and increment the FIFO counter. 'Reads' will read from the head of the FIFO as the shift register would in normal mode. The FIFO count will be decremented on reads. Aside from reading data with the CPU rather than the shift register, the FIFO and all associated control, status and interrupt registers work the same as in normal mode. When the FIFO is software visible, the transmit shift register cannot access the FIFO.

The maximum depth of the FIFO is indicated by the TFFMAX register. If ffEnable (TFFCR[1]) is set to zero, the FIFO is disabled. If the FIFO is disabled, the depth of the FIFO is one, regardless of the depth that the TFFMAX register indicates.

Each write to the FIFO will write an additional entry in the FIFO and increment the TFFCNT register. The FIFO will overflow if the FIFO is written when full. The FIFO is full if the values in the TFFCNT and TFFMAX registers are equal, or if ffEnable (TFFCR[1]) indicates the FIFO is disabled and the value in the TFFCNT register is '1.'

If the FIFO overflows, the last entry in the FIFO will be overwritten. The tffError (TFFHR[8]) bit for that entry in the FIFO will be set to one, regardless of the value that was written to that bit by the CPU.

5.2.15.1.2 SUCGPU_RFFHR/SUCSCO_RFFHR Registers.

These registers are used to access the UART receive FIFO. In normal mode, the FIFO is readable by the CPU, but not writable. The FIFO data is written by the receive shift register when it receives a full character, serially transmitted on the UART_RXD/SMC_DATA signal.

Table 5-203 SUCGPU_ RFFHR/SUCSCO_RFFHR Bit Definitions (0x0400_A040,0x0400_A840)

Bits	Symbol	Meaning
31:9	RESERVED	Reserved for future use
8	rffError	Receive FIFO Error. The rffError field is ignored for transmission. It is not read by the receive shift register. If the FIFO is software visible, bit 8 of the CPU write data will be written to the FIFO, unless the write causes a FIFO overflow, in which case it is forced to a '1.' State after reset is 0x00. A software FIFO reset, writing a '1' to ffReset (RFFCR[0]) will set all rffError bits in the FIFO to '0.'
7:0	rffData	Receive FIFO Data. The rffData[7:0] field is the data that was serially shifted into the receive shift register. All entries are set to 0x00 on reset. The state is not affected by a software FIFO reset, writing a '1' to ffReset (RFFCR[0]).

When ffSWVisible (RFFCR[2]) is set to '1,' the FIFO is readable and writable from the CPU. 'Write' operations write to the tail of the FIFO and increment the FIFO counter as the receive shift register would. 'Reads' will read from the head of the FIFO as the CPU would in normal mode. Reads will decrement the FIFO counter. Aside from writing data with the CPU rather than the shift register, the FIFO and all associated control, status and interrupt registers work the same as in normal mode. When the FIFO is software visible, the receive shift register cannot access it.

The maximum depth of the FIFO is indicated by the RFFMAX register. If ffEnable (RFFCR[1]) is set to '0,' the FIFO is disabled. If the FIFO is disabled, the depth of the FIFO is '1' regardless of the depth that the RFFMAX register indicates.

Each write to the FIFO will write an additional entry in the FIFO and increment the RFFCNT register. The FIFO will overflow if the FIFO is written when full. The FIFO is full if the RFFCNT and RFFMAX registers are equal or if ffEnable (RFFCR[1]) indicates the FIFO is disabled and RFFCNT is '1.'

If the FIFO overflows, the last entry in the FIFO will be overwritten. The rffError (RFFHR[8]) bit for that entry in the FIFO will be set to '1' regardless of the value that was written to that bit by the CPU.

5.2.15.1.3 SUCGPU_TFFHRSRW/ RFFHRSRW, SUCSCO_TFFHRSRW/ RFFHRSRW Registers

These registers are for debugging and testing only. They function exactly as their FFHR counterpart registers (see the previous register description). The only difference is that any reads or writes to these registers will force a simultaneous read or write to the FIFO. On writes, the read data is ignored. On reads, the write data is whatever value is on the *bus_data* bus at the time of the write.

Table 5-204 SUCGPU_ TFFHRSRW/RFFHRSRW, SUCSCO_TFFHRSRW/RFFHRSRW Bit Definitions (0x0400_A004, 0x0400_A804 and 0x0400_A044, 0x0400_A844)

Bits	Symbol	Meaning
31:9	RESHRVED	Reserved for future use
8	ffError	Transmit/Receive FIFO Error. This field accesses the same registers as the tffError and rffError fields in the TFFHE and RFFHR registers. See the TFFHE/RFFHR register description for details. State after reset is 0x00. A software FIFO reset, writing a '1' to ffReset (TFFCR[0]) will set all tffError bits in the fifo to '0.'
7:0	ffData	Transmit/Receive FIFO Data. This field accesses the same registers as the tffData and rffData fields in the TFFHE and RFFHR registers. See the TFFHE/RFFHR register description for details. All entries are set to 0x00 on reset. The state is not affected by a software FIFO reset, writing a '1' to ffReset (TFFCR[0]).

Note: In SOLO1 Rev.1.x, these registers are accessible in normal mode, so care must be taken to ensure that they are not written or read unintentionally. Doing so will change the contents of the FIFOs.

5.2.15.2 SUCGPU_TFFTRG /SUCSCO_TFFTRG Registers

These registers specify the transmit FIFO trigger level to indicate interrupts and status bits to the CPU. The FIFO compares the current number of FIFO entries tffCount (TFFCNT) to the value specified in TFFTRG.

The transmit FIFO trigger should be used to indicate when the number of entries in the FIFO drops below the specified level. If the Interrupt Enable register (IER) is set appropriately, the CPU will be interrupted so that the FIFO can be filled as needed.

The tffLteTrigger (TFFSR[0]) and tffGteTrigger (TFFSR[1]) status bits indicate whether the current count, tffCount (TFFCNT), is "less than or equal to" or "greater than or equal to" the specified trigger, respectively.

A transmit FIFO interrupt, tLteTrigger (ISR[4]) will be issued if the tffCount (TFFCNT) is less than or equal to tffTrigger (TFFTRG).

If the transmit FIFO is disabled, with ffEnable (TFFCR[1]), the trigger value should be set to '0' if the FIFO is to be used in as a one-deep FIFO in normal mode.

Table 5-205	SUCGPU_	TFFTRG/SUCSCO_	TFFTRG Bit	Definitions
	$(0x0400_A$	008,0x0400_A808)		•

Bits	Symbol	Meaning
31:8	Zero	Will always read 0x0.
7:0	tffTrigger	Transmit FIFO Trigger. This field specifies the trigger point in the transmit FIFO. The value of the trigger should always be written so that it is between '0' and the value specified in tffMaxDepth (TFFMAX), inclusive, if the FIFO is enabled. If the FIFO is used when disabled, the trigger should be set to '0.' State after reset is 0x00.

5.2.15.2.1 SUCGPU_RFFTRG /SUCSCO_RFFTRG Registers

These registers specify the receive FIFO trigger level to indicate interrupts and status bits to the CPU. The FIFO compares the current number of FIFO entries rffCount (RFFCNT) to the value specified in RFFTRG.

The receive FIFO trigger should be used to indicate with the number of entries in the FIFO drops below the specified level. If the Interrupt Enable register (IER) is set appropriately, the CPU will be interrupted so that the FIFO can be emptied.

The rffLteTrigger (RFFSR[0]) and rffGteTrigger (RFFSR[1]) status bits indicate whether the current count, rffCount (RFFCNT), is "less than or equal to" or "greater than or equal to" the specified trigger, respectively.

A receive FIFO interrupt, rLteTrigger (ISR[2]) will be issued if the rffCount (RFFCNT) is less than or equal to rffTrigger (RFFTRG).

If the receive FIFO is disabled, with ffEnable (RFFCR[1]), the trigger value should be set to '0' if the FIFO is to be used in as a one-deep FIFO in normal mode.

Table 5-206 SUCGPU_RFFTRG/SUCSCO_RFFTRG Bit Definitions (0x0400_A048,0x0400_A848)

Bits	Symbol	Meaning
31:8	Zero	Will always read 0x0.
31:0	rffTrigger	Receive FIFO Trigger. This field specifies the trigger point in the receive FIFO. The value of the trigger should always be written so that it is between '0' and the value specified in rffMaxDepth (RFFMAX), inclusive, if the FIFO is enabled. If the FIFO is used when disabled, the trigger should be set to '0.' State after reset is 0x0.

The receive trigger is used to drive the UART_RTS (Request to Send) signal when under hardware control. Hardware control of UART_RTS is specified by setting the spioEnableSCRRTS_N (SPIOEN[3]) bit in the SUCGPU/SUCSCO_SPIOEN register (refer to Table 5-236). Under hardware control, UART_RTS is asserted when the rffCurCount (RFFCNT) is greater than or equal to rffTrigger (RFFTRG).

5.2.15.2.2 SUCGPU_TFFCNT /RFFCNT, SUCSCO_TFFCNT /RFFCNT Registers

These registers indicate the current number of entries in the FIFO. Writes to the FIFO increment the count. Reads from the FIFO decrement the count.

Table 5-207 SUCGPU_TFFCNT/RFFCNT and SUCSCO_TFFCNT/RFFCNT Bit Definitions (0x0400_A00C,0x0400_A04C and 0x0400_A80C,0x0400_A84C)

Bits	Symbol	Meaning
31:8	Zero	Will always read 0x0.
7:0	ffCount	FIFO Current Count The ffCount field indicates the current number of entries in the FIFO. The value of ffCount is will always be between '0' and the value of ffMaxDepth (FFMAX) if the FIFO is enabled. If the FIFO is disabled, ffCount will always be either '0' or '1.' State after reset is 0x0.

5.2.15.2.3 SUCGPU_TFFMAX/RFFMAX, SUCSCO_TFFMAX/RFFMAX Registers

These registers are constant read-only registers that indicate the maximum depth of the FIFO. Software can read these registers to determine the FIFO's depth. If the transmit and receive FIFOs are swapped, through ffSwap (FFGCR[0]), the FFMAX value will also be swapped. Software can always depend on the FFMAX register reading the maximum depth of the FIFO.

Note that the FFMAX registers are unaffected by ffEnable (FFCR[1]). They will read the maximum FIFO depth available if the FIFO was enabled. If not enabled, the FIFO depth will effectively be '1,' but the FFMAX registers will not reflect this.

Table 5-208 SUCGPU_TFFMAX/RFFMAX and SUCSCO_TFFMAX/RFFMAX Bit Definitions (0x0400_A010, 0x0400_A050 and 0x0400_A810, 0x0400_A850)

Bits	Symbol	Meaning
31:8	Zero	Will always read zero
7:0	ffDepth	FIFO Maximum Depth. The ffCount field indicates the maximum number of entries in the FIFO. State after reset is the depth of the FIFO.

5.2.15.2.4 SUCGPU_TFFCR/RFFCR, SUCSCO_TFFCR/RFFCR Registers

These registers control the transmit and receive FIFOs.

Table 5-209 SUCGPU_TFFCR/RFFCR and SUCSC0_TFFCR/RFFCR Bit Definitions (0x0400_A014, 0x0400_A054 and 0x0400_A814, 0x0400_A854)

Bits	Symbol	Meaning
313	RESERVED	Reserved for future use.
2	ffSWVisible	FIFO Software Visible. The ffSWVisible bit makes the FIFO readable and writable through the FFHR register. Normally the transmit FIFO is only writable and the receive FIFO is only readable. When the ffSWVisible bit is set to '1,' the transmit and receive shift registers cannot access the FIFOs. With the ffSWVisible bit set, all software read and writes to the FIFOs will affect the status, control and interrupt registers as if the FIFO had been read or written in normal mode. State after reset is 0x0.

Table 5-209 SUCGPU_TFFCR/RFFCR and SUCSCO_TFFCR/RFFCR Bit Definitions (0x0400_A014, 0x0400_A054 and 0x0400_A814, 0x0400_A854) (Continued)

Bits	Symbol	Meaning
1	ffEnable	FIFO Enable. The ffEnable bit is used to enable FIFO mode. Writing a '1' will enable the FIFO, writing a '0' will disable it. A disabled FIFO will always have an effective depth of one, regardless of the value of ffMaxDepth (FFMAX). Enabling or disabling the FIFO has no effect of ffMaxDepth (FFMAX). The ffEnable bit has no effect on FIFOs that are one byte deep as specified by ffMaxDepth (FFMAX). The ffEnable bit does not affect the trigger level, ffTrigger (FFTRG). Software should set the trigger to zero if the FIFO is to be used as a FIFO of depth one when disabled. Disabling the FIFO by clearing the ffEnable bit will force a reset of the FIFO as if the ffReset bit had been written with a '1.' This guarantees that the FIFO count, ffCount (FFCNT) will always be a legal value. State after reset is 0x0.
0	ffReset	FIFO Software Reset. Writing a '1' to the ffReset bit clears all the error bits in the FIFO and resets the ffCount (FFCNT) register to zero. The actual data in the FIFO is not affected, but the FIFO is effectively emptied, since the FIFO pointers are reset. Writing a '0' to the ffReset bit has no effect. The reset will be complete one cycle after the ffReset bit is written. It is not necessary to read the ffReset bit to determine when the FIFO has been reset. The ffReset bit is self clearing and will always read zero. State after reset is 0x0.

5.2.15.2.5 SUCGPU_TFFSR/RFFSR, SUCSCO_TTFFSR/RFFSR Registers

These registers contain read-only FIFO status.

Table 5-210 SUCGPU_ TFFSR/RFFSR and SUCSCO_ TFFSR/RFFSR Bit Definitions (0x0400_A018, 0x0400_A058 and 0x0400_A818, 0x0400_A858)

Bits	Symbol	Meaning
31:4	RESERVED	Reserved for future use:
3	ffAnyErrorInFifo	Any Error In FIFO. This bit is set to '1' if any of the error bits associated with the FIFO entries is set to '1.' State after reset is 0x0.
2	ffFifoOverflow	FIFO Overflow. This bit is set to '1' if a write to the FIFO occurred when the FIFO was full (FFCNT equal to FFMAX). State after reset is 0x0.
1	ffGteTrigger	Trigger Count Greater-than or Equal-to Trigger. This bit is set to '1' if, and only if, the number of entries in the FIFO, ffCount (FFCNT), is greater-than or equal-to the trigger for the FIFO, ffTrigger (FFTRG). State after reset is 0x1.
0	ffLteTrigger	Trigger Count Less-than or Equal-to Trigger. This bit is set to '1' if, and only if, the number of entries in the FIFO, ffCount (FFCNT), is less than or equal to the trigger for the FIFO, ffTrigger (FFTRG). State after reset is 0x1.

5.2.15.2.6 SUCGPU_TFFGCR/RFFGCR, SUCSCO_TFFGCR/RFFGCR Registers

These registers contain control bits that affect both the receive and transmit FIFOs. Note that the TFFGCR and RFFGCR registers always access the same register.

Table 5-211 SUCGPU_TFFGCR/RFFGCR and SUCSCO_TFFGCR/RFFGCR Bit Definitions (0x0400_A01C, 0x0400_A05C and 0x0400_A81C, 0x0400_A85C)

Bits	Symbol	Meaning
31:1	RESERVED	Reserved for future use.
0	ffSwap	FIFO Swap. If the ffSwap bit is set, the receive and transmit FIFOs are swapped. This should be almost transparent to software (assuming the FIFOs were empty when swapped). That is, all transmit FIFO register accesses should still be done with the TFF* registers, and all receive FIFO accesses should be done with the RFF* registers. The only difference will be that the max size of the receive and transmit FIFOs will be swapped. Software can always read the FFMAX register to determine the current size of the FIFO. If the receive and transmit FIFOs have different depths, software can use the ffSwap bit to select which FIFOs should be used for receive and transmit. Software can use this feature to minimize interrupt overhead, based on whether more data is being transmitted or received.

5.2.15.3 Shift Register Registers

5.2.15.3.1 SUCGPU_TSRCR/SUCSC0_TSRCR Registers.

This register controls the operation of the transmit shift register.

Table 5-212 SUCGPU_TSRCR/SUCSCO_TSRCR Bit Definitions (0x0400_A080, 0x0400_A880)

Bits	Symbol	Meaning
31:2	(RESERVED)	Reserved for future use
1	tsrCTSFlowCntl	Enable HW CTS Flow Control If set, this bit enables the hardware flow control mechanism. If enabled, the Transmit Shift register will not be loaded with new data if the UART_CTS_N (Clear-to-Send) signal is deassert- ed, effectively disabling the Transmit Shift regis- ter. This allows the receiving device to throttle the transmitter without software intervention. State after reset is 0x0.
0	tsrEnable	Enable Transmitter Shift Register If this bit is set, the Transmit Shift register is loaded with read data from the transmit FIFO, assuming that the shift register is empty and that the FIFO has data available. The data is shifted out serially on the transmit data line (UART_TXD/SMC_DATA). Once the data has been loaded into the shift register, it is shifted out, regardless of the state of the tsrEnable bit. If the tsrEnable bit is cleared, it will prevent the next data in the FIFO from being loaded into the shift register. State after reset is 0x00.

Smart Cards require parity detection of transmit data. If a transmit error is detected by the Smart Card, it will pull the 'Stop' bit low. If such a condition is detected, the Transmit Shift register will be disabled (although the tsrEnable bit will still be set), and the tParityErrDetect (ISR[6]) bit will indicate that this situation occurred. Writing a '1' to the tsrEnable bit (whether it is already a '1' or not), will clear the tParityErrDetect bit and re-enable the Transmit Shift register.

5.2.15.3.2 SUCGPU_RSRCR/SUCSC0_RSRCR Register

This register provides the receive shift register control.

Table 5-213 SUCGPU_RSRCR/SUCSC0_RSRCR Bit Definitions (0x0400_A0C0, 0x0400_A8C0)

Bits	Symbol	Meaning
31:2	RESERVED	Reserved for future use.
1	rsrFiltDisable	Disable Receive Filter If this bit is set, the receive data filter will be disabled. If this bit is cleared, the filter will be enabled. If the filter is enabled, receive data will be sample three times. The receive data is sampled on the system clock if sampDataSysclk (CCR[3]) is set, otherwise, data is sampled on the master clock. Note that on SOLO1 1.x chips, sampDataSysclk (CCR[3]) must be set to '0' for proper receive operation. The Start bit will be detected until three consecutive samples are low. If the Start bit does not remain low for half a bit time, the Start bit is assumed to be erroneous and is ignored. This is intended to filter out spurious transitions on the receive data line. State after reset is 0x0.
0	rsrEnable	Enable Receiver Shift Register If this bit is set, the Receive Shift register will detect serial data on the receive signal (UART_RXD/ SMC_DATA). Once the data has been received, it will be loaded into the receive FIFO. If the rsrEnable is cleared, any receive data will be ignored. State af- ter reset is 0x0.

5.2.15.3.3 SUCGPU_TSRSTATE /SUCSCO_TSRSTATE Registers

These registers are read-only and always read the state of the transmit shift register state machine.

Table 5-214 SUCGPU_TSRSTATE/SUCSCO_TSRSTATE Bit Definitions (0x0400_A084, 0x0400_A884)

Bits	Symbol	Meaning
31:4	RESERVED -	Reserved for future use.
3:0	tsrState	State of Transmit Shift State Machine This field indicates the current state of the transmitter state machine. The states indicate which part of the character is a currently being transmitted on the transmit data signal. The transmitter states are encoded as follows: Ob0000 TSM_IDLE Ob0001 TSM_START Ob0010 TSM_DATA Ob0100 TSM_DATA Ob0100 TSM_PARITY Ob1000 TSM_STOP State after reset is 0b0000.

5.2.15.3.4 SUCGPU_RSRSTATE /SUCSCO_RSRSTATE Registers

These registers are read-only and contain the receive shift register state.

Table 5-215 SUCGPU_RSRSTATE/SUCSCO_RSRSTATE Bit Definitions (0x0400_A0C4, 0x0400_A8C4)

Bits	Symbol	Meaning
31.4	RESERMED	Reserved for future use
3:0	rsrState	State of Receive Shift State Machine This field indicates the current state of the receiver state machine. The states indicate which part of the character is a currently being received on the receive data signal. The receiver states are encoded as follows: 0b0000 RSM_IDLE 0b0001 RSM_START 0b0010 RSM_DATA 0b0100 RSM_PARITY 0b1000 RSM_STOP 0b1100 RSM_STOP 0b1100 RSM_DRVPAR State after reset is 0b0000.

5.2.15.3.5 SUCGPU_TSRBCCNT /SUCSCO_TSRBCCNT Registers

These registers are read-only and contain the current state of the transmit bit clock counter.

Table 5-216 SUCGPU_TSRBCCNT/SUCSCO_TSRBCCNT Bit Definitions (0x0400_A088, 0x0400_A888)

Bits	Symbol	Meaning
31:4	RESERVED	Reserved for future use
3:0	tsrBCCnt	Transmit Bit Clock Count This field always contains the current transmit bit clock count. For each bit that is transmitted, the bit clock counter counts down 16 sample clock ticks. This count indicates how many sample clocks are left in the current bit being transmitted. State after reset is 0x0.

5.2.15.3.6 SUCGPU_RSRBCCNT /SUCSCO_RSRBCCNT Registers

These registers are read-only and contain the current state of the receive bit clock counter.

Table 5-217 SUCGPU_ RSRBCCNT/SUCSCO_ RSRBCCNT Bit Definitions (0x0400_A0C8, 0x0400_A8C8)

Bits	Symbol	Meaning
314	RRESIDENTED	Reserved for future use
3:0	rsrBCCnt	Receive Bit Clock Count This field always contains the current receive bit clock count. With the exception of the Start bit, the bit clock counter counts down 16 sample clock ticks for each bit that is received. The counter only counts down 8 sample clock ticks for the Start bit, so that the receive data is sampled in the nominal center of the receive data bits. This count indicates the how many sample clocks are left in the current bit being received. State after reset is 0x0.

5.2.15.3.7 SUCGPU_TSRBITCNT/SUCSCO_TSRBITCNT Registers

These registers are read-only and contain the current state of the transmit data bit counter.

Table 5-218 SUCGPU_TSRBITCNT/SUCSCO_TSRBITCNT Bit Definitions (0x0400_A08C, 0x0400_A88C)

Bits	Symbol	Meaning
31:4	RESERVED	Reserved for future use.
3:0	tsrBitCnt	Transmit Current Bit Count This field always contains the number of bits left to transmit of the current character in the transmit shift register. The count reflects the number of bits left to transfer after the current bit has been completed. For example, if the second bit of a 7-bit character is being transferred, the tsrBitCnt will read 0x5. The tsrBitCnt will read as 0x0 if no character is being transmitted. State after reset is 0x0.

5.2.15.3.8 SUCGPU_RSRBITCNT /SUCSCO_RSRBITCNT Registers

These registers are read-only and contain the current state of the receive data bit counter.

Table 5-219 SUCGPU_TSRBITCNT/SUCSCO_TSRBITCNT Bit Definitions (0x0400_A0CC, 0x0400_A8CC)

Bits	Symbol	Meaning
315	RESERVED	Reserved for future use
2:0	rsrBitCnt	Receive Current Bit Count This field always contains the number of bits left to receive of the current character being transferred into the receive shift register. The count reflects the number of bits left to transfer after the current bit has been completed. For example, if the second bit of a 7-bit character is being transferred, the rsrBitCnt will read 0x5. rsrBitCnt will read as 0x0 if no character is currently being received. State after reset is 0x0.

5.2.15.4 Clock Divider Registers

5.2.15.4.1 SUCGPU_MCD/SUCSCO_MCD Registers.

These registers specify the divisor from the system clock that is used to generate the master clock for asynchronous Smart Card applications. This divided clock is driven onto the SMC_CLK signal. It may also be used as the sample clock for the receive data filter, if enabled by the sampDataSysclk bit (CCR[2]). Other than that, the master clock is not used in UART mode.

Note: On SOLO1 Rev. 1.x chips, the master clock must be enabled for proper operation in UART mode. Setting MCD to '0' for the UART case ensures proper operation.

Table 5-220 SUCGPU_ MCD0/SUCSC0_ MCD0 Bit Definitions (0x0400_A100, 0x0400_A900)

Bits	Symbol	Meaning
31:6	RESERVED	Reserved for future use
5:0	mastClkDiv	Master Clock Divisor This field specifies the divisor to generate the master clock from the system clock. The clock will toggle every mastClkDiv+1 system clock cycles. The system clock will be divided by (mastClkDiv+1)*2. For example, if the system clock is 83MHz (12ns period) and the mastClkDiv is set to 10, the master clock will be a 83/((10+1)*2) = 3.77MHz clock. Smart Card master clocks will generally want to be somewhere between 1MHz and 5MHz. State after reset is 0x0.

5.2.15.4.2 SUCGPU_SCD0/SCD1, SUCSC0_SCD0/SCD1 Registers

These registers are used to specify a divisor in order to generate the UART sample clock from the system clock. The sample clock is used to generate the sample points for receive data and to generate the proper bit timing for transmit data. The sample clock is divided by 16 to generate the bit clock.

Table 5-221 SUCGPU_ SCD0/SCD1, SUCSC0_ SCD0/SCD1 Bit Definitions (0x0400_A104, 0x0400_A108, 0x0400_A904, 0x0400_A908)

Bits	Symbol	Meaning
-31:13 = -	RESERVED	Reserved for future use.
12:0	sampClkDiv	Sample Clock Divisor The sampClkDiv field is a 13-bit field generated by using SCD0 [7:0] as the least significant bits and SCD1[4:0] as the most significant bits. The sam- ple clock is generated by dividing the system clock by sampClkDiv + 1.

Table 5-221 SUCGPU_ SCD0/SCD1, SUCSC0_ SCD0/SCD1 Bit Definitions (0x0400_A104, 0x0400_A108, 0x0400_A904, 0x0400_A908) (Continued)

	0x0400_A108, 0x0400_A304, 0x0400_A308/ (Continued)		
Bits	Symbol	Meaning	
	sampClkDiv (continued)	For example, if sampClkDiv is specified as 0x0, the sample clock will be the same as the system clock. A sampClkDiv value of 0x1f would specify a clock that is 32 times slower than the system clock. The sample clock is always defined to be 16 times the bit rate of the serial transfer. So, to set up a particular baud rate for a UART, the sampClkDiv field should be computed as follows: f(SYS_2XCKIN) sampClkDiv =	
		Where f (SYS_2XCLKIN) is the system frequency and BAUDRATE is the target baud rate. For example, to program a system running at 83MHz for 9600 baud character transfer, the sampClkDiv register should be programmed to 539. Setting the sample clock for asynchronous Smart Cards is a little more difficult. The sample clock must have a specific relationship with the master clock for correct functionality. The details and more examples are described in the "Choosing Smart Card Clock Divisors" section. But for the answer to reset and most common Smart Cards, the clock divisor should be set up so all the following equations are satisfied:	
		f(SYS_2XCLKIN) f(SMC_CLK) =	
		1 MHz <= f(SMC_CLK) <= 5 MHz	
		(mastClkDiv + 1) * 2 * 372 sampClkDiv =	
		mastClkDiv must be odd (since sampClkDiv must be an exact integer).	

 Table 5-221
 SUCGPU_ SCD0/SCD1, SUCSC0_ SCD0/SCD1
 Bit Definitions (0x0400_A104, 0x0400_A904, 0x0400_A908) (Continued)

Bits	Symbol	Meaning
12:0	sampClkDiv (continued)	Effective Baud rate =
		For example, if the system frequency, f(SYS_2XCLKIN), is 83MHz, mastClkDiv should be 11, in order to generate a Smart Card clock, f(SMC_CLK) of 3.46MHz. The sampClkDiv would then be 557. This gives an effective baud rate of 9297 bits/s. State after reset is 0x0.

5.2.15.4.3 SUCGPU_CCR/SUCSC0_CCR Register

These registers control the Smart Card master clock and the sample clock.

Table 5-222 SUCGPU_ CCR/SUCSCO_CCR Bit Definitions (0x0400_A10C, 0x0400_A90C)

Bits	Symbol	Meaning
.31:3	SKRSERVIED -	Reserved for future use
2	sampDataSysClk	Sample Data on System Clock If sampDataSysclk is set, the sample clock for the receive data filter is the system clock. If sampDataSysclk is cleared, the sample clock will be the master clock. See the definition of rsrFiltDisable for more information. State after reset is 0x0. Note that on SOLO1 1.x chips, the sampData- SysClk field must be set to '0' for proper oper- ation.

Table 5-222 SUCGPU_ CCR/SUCSCO_CCR Bit Definitions (0x0400_A10C, 0x0400_A90C) (Continued)

Bits	Symbol	Meaning
1	sampClkEn	Sample Clock Enable If this bit is set, the sample clock counter continually counts down the value of sampClkDiv (SCD1 [4:0], SCD0 [7:0]). If sampClkEn is cleared, the counter is immediately frozen. State after reset is 0x0.
0	mastClkEn	Master Clock Enable If this bit is set, the master clock counter continually counts down the value of mastClkDiv (MCD[5:0]). If mastClkEn is cleared, the counter is frozen by not reloading the mast-ClkDiv value when the counter reaches zero. When the counter is disabled it will not generate a short pulse. The master clock is always low when disabled. State after reset is 0x0.

5.2.15.5 Line Control and Status Registers

5.2.15.5.1 SUCGPU_LCR/SUCSCO_LCR Registers

These registers control the operation of the transmit and receive data lines. They control the word lengths, parity logic and the send-break mechanism.

Table 5-223 SUCGPU_LCR/SUCSCO_LCR Bit Definitions (0x0400_A180, 0x0400_A980)

Bits	Symbol	Meaning
31:7	RESERVED :	Reserved for futurense.
6	setBreak	Set Break If this bit is set, a break condition is transmitted on the transmit data line. The break condition is generated by forcing the serial transmit data to the logic '0' (spacing) state. The break is disabled by clearing the setBreak bit. setBreak only affects the value on the transmit data signal. The rest of the transmit logic behaves normally. State after reset is 0x0.

Table 5-223 SUCGPU_LCR/SUCSCO_LCR Bit Definitions (0x0400_A180, 0x0400_A980) (Continued)

Bits	Symbol	Meaning
5	stickParity	Stick Parity When the stickParity bit is set, a constant value is forced as parity. When parityEn, evenParity and stickParity are all set, the parity bit is transmitted and checked as a logical '0.' When parityEn and stickParity are set, but evenParity is cleared, the parity bit is transmitted and checked as a logical '1.' If the stickParity bit is cleared, stick parity is disabled. State after reset is 0x0.
4	evenParity	Even Parity When evenParity and parityEn are set, an even number of logical '1's are transmitted or checked in the data word bits and parity bit. If evenParity is cleared and parityEn is set, an odd number of logical '1's are transmitted or checked. State after reset is 0x0.
3	parityEn	Parity Enable When parityEn is set, it indicates that a parity bit is generated (transmit data) or checked (receive data) between the last data word bit and the stop bit of the serial data. The parity bit is used to produce an even or odd number of '1's when the data word bits and the parity bit are summed. When parityEn is cleared, no parity bit is generated/checked. State after reset is 0x0.
1:0	RESERVED wordLength	Word Length in Bits The wordLength field indicates the number of bits in each character that is transmitted or received. The encoding is as follows: wordLength Character Length 0x0 5 bits 0x1 6 bits 0x2 7 bits 0x3 8 bits State of for receiving OxO
		State after reset is 0x0.

5.2.15.5.2 SUCGPU_LSCR/SUCSC0_LSCR Registers

These registers controls the transmit and receive lines for functions specific to Smart Card operation.

Table 5-224 SUCGPU_LSCR/SUCSC0_LSCR Bit Definitions (0x0400_A184, 0x0400_A984)

Bits	Symbol	Meaning
31:2	RESERVED	Reserved for future use.
2	parityNotify	Check Smart Card Parity When this bit is set, the receiver will signal a parity error to the transmitting device if such an error is detected. The parity error is signalled by pulling the receive data line low during the Stop bit of the char- acter that is being received. The transmitting device will detect this and reissue the character transmis- sion for up to three characters. This only makes sense for use with asynchronous Smart Cards (which have the data signal implemented as open drain). In normal operation, the parityNotify bit should be cleared and the transmitting device will not be notified of the error. State after reset is 0x0.
1	invertBitPol	Invert bit Polarity Setting this bit inverts the logical sense of each bit that is transmitted or received. For normal operation, the invertBitPol bit should be cleared. In order to use the inverse data transfer mode for Smart Cards that require it, the swapBitOrder and invertBitPol bits should both be set to '1.' State after reset is 0x0.
0	swapBitOrder	Swap the Transmit/Receive Bit Order Setting this bit reverses the order in which the bits of the character are shifted out. For normal opera- tion, the swapBitOrder bit is cleared so that the least significant bit will be transmitted and re- ceived first. If the swapBitOrder bit is set, the most significant bit will be transmitted/received first. In order to use the inverse data transfer mode for Smart Cards that require it, the swapBitOrder and invertBitPol bits should both be set to '1.' State after reset is 0x0.

5.2.15.5.3 SUCGPU_LSTPBITS/SUCSCO_LSTPBITS Registers

These registers specify the number of stop bits that should be transmitted after each character. This does not affect the receiver since it always looks only at the first stop bit. Smart Cards may request up to 256 stop bits after each character that is transmitted.

Table 5-225 SUCGPU_LSTPBITS/SUCSC0_LSTPBITS Bit Definitions (0x0400_A188, 0x0400_A988)

Bits	Symbol	Meaning
31:8	RESERVED	Reserved for future use.
7:0	stopBits	Number of Stop Bits The stopBits field is used to indicate the number of Stop bits that the transmitter should send before starting the next character transmission. The number of Stop bits transmitted will be stopBits + 1.

5.2.15.5.4 SUCGPU_LSR/SUCSCO_LSR Registers

These registers contain current status of the transmit and receive data transfers. The overrunError, parityError, framingError and breakDetect bits provide imprecise error information. If any of these errors are encountered, the appropriate bit in the LSR register is immediately set. One error bit is also associated with the entry in the FIFO that caused the error (i.e., the rffError bit (RFFHR[8])). When software reads that data, it determines that there is an error condition and reads the LSR to determine the cause. The exact cause may not be determined if there is more than one error condition flagged in the FIFO. These error bits are only cleared if the LSR is read when there are no more error bits set in the FIFO.

Multiple errors are expected to be pretty rare. And if they exist, software will likely have to handle the situation though higher-level protocols anyway.

Table 5-226 SUCGPU_LSR/SUCSCO_LSR Bit Definitions (0x0400_A18C, 0x0400_A98C)

Bits	Symbol	Meaning
91.8	RESERVED	Reserved for future use
7	errorInRFifo	Any error in the receive FIFO When set, this bit indicates that one of the entries in the receive FIFO has an error associated with it. If this bit is set, at least one of the overrunError, parityError, framingError, or breakDetect bits will be set. This bit is cleared by resetting the receive FIFO or by reading all of the entries in the receive FIFO.

Table 5-226 SUCGPU_LSR/SUCSCO_LSR Bit Definitions (0x0400_A18C, 0x0400_A98C) (Continued)

Bits	Symbol	Meaning
6	transmitter- Empty	Shift register empty When set, the transmitterEmpty bit indicates that the last character of the data in the transmit shift register has been transmitted and the transmit FIFO is empty. This can be used to determine when all data has been transmitted. This status also avail- able as an interrupt in the ISR, (transmitterEmpty (ISR[2]).
5	tFifoEmpty	Transmit FIFO empty When set, this bit indicates that the transmit FIFO is empty. This means that the last data has been trans- ferred out of the FIFO into the transmit shift regis- ter. This bit is cleared by writing data to the FIFO. If the transmitter is empty, it may only be cleared mo- mentarily.
4	breakDetect	Break detect in the receive FIFO. When set, this bit indicates that a break detect was received on one of the entries in the fifo. The FIFO entry will have a set error bit associated with it. Once the breakDetect bit is set, it will only be cleared when there are no more errors in the FIFO (errorInRFifo is cleared) and the LSR has been read.
3	framingError	Framing error in the receive FIFO. When set, the framingError bit indicates that a framing error was received on one of the entries in the FIFO. The FIFO entry will have a set error bit associated with it. Once the framingError bit is set, it will only be cleared when there are no more errors in the FIFO (errorInRFifo is cleared) and the LSR has been read.
2	parityError	Parity error in the receive FIFO. When set, this bit indicates that a parity error was received on one of the entries in the FIFO. The FIFO entry will have an error bit associated with it. Once the parityError bit is set, it will only be cleared when there are no more errors in the FIFO (error-InRFifo is cleared) and the LSR has been read.

Table 5-226 SUCGPU_LSR/SUCSCO_LSR Bit Definitions (0x0400_A18C, 0x0400_A98C) (Continued)

Bits	Symbol	Meaning
1	overrunError	Overrun error in the receive FIFO. When set, the overrunError bit indicates that the FIFO was full when another character was received. The last entry in the FIFO was overwritten when the overflow occurred. If the FIFO was disabled, it is treated as if it is only one entry deep and the only entry was overwritten. Once the overrunError bit is set, it will only be cleared when there are no more errors in the FIFO (errorInRFifo is cleared) and the LSR has been read.
0	dataReady	Receive data ready When set, this bit indicates that an input character has been received and transferred to the FIFO. This is essentially an indicator of whether or not the re- ceive FIFO is empty or not. The dataReady bit is reset to zero by reading all of the data in the receive FIFO or by resetting the receive FIFO.

5.2.15.6 Interrupt Registers

5.2.15.6.1 SUCGPU_ISR/ISR_R, SUCSCO_ISR/ISR_R Registers

These registers contain the status of the sucUnit interrupts. Each bit in the register corresponds to a particular interrupt. If the bit is set, that means that the interrupt condition is true and that the CPU will be interrupted if the corresponding bit is set in the interrupt enable register (IER). If the bit is cleared, the interrupt condition is not currently pending.

The ISR registers always contain the current interrupt information masked by the corresponding bits in the IER registers. If the corresponding bit in the IER is '0,' the ISR register will always read '0' for that bit. If the bit in the IER register is '1,' the ISR will contain a '1' if the interrupt is pending, a '0' if it is not.

In normal operation, the interrupt status registers are read-only. They must be cleared by performing the actions required to deassert the interrupts. Software can ignore pending interrupts by disabling the interrupts in the IER register.

The interrupt status registers are also writable, but this should only be used for chip-level diagnostics to test the data path to the registers. These registers should never be written during normal operation.

Table 5-227 SUCGPU_ ISR/ISR_R, SUCSCO_ ISR/ISR_R Bit Definitions (0x0400_A200, 0x0400_AA00 and 0x0400_A204, 0x0400_AA04)

Bits	Symbol	Meaning
31:8	RESERVED	Reserved for future use.
7	gpioInterrupt	GPIO Interrupt When set, the gpioInterrupt field indicates that a GPIO interrupt is enabled and pending. The GPIO interrupts are another level of interrupts. This bit indicates that at least one of the interrupts at the GPIO level is pending. The state of the GPIO inter- rupts can be determined by reading the GPIOISR registers. To clear the interrupt, all the GPIO inter- rupts must be cleared or disabled.
6	tParityErr Detect	Transmit Parity Error When set, the tParityErrDetect field indicates that a Smart Card device detected a parity error on data that was transmitted to the card. When this condition is detected, the transmitter is disabled. Writing a '1' to the tsrEnable bit of the TSRCR register (regardless if it is already a one or not), will clear the tParityErrDetect interrupt and re-enable the transmit shift register.
5	transmitter- Empty	Transmit Shift Register Empty When set, the transmitterEmpty field indicates that the transmit FIFO is empty and that the last bit in the transmit shift register has been transmitted. To clear this interrupt, data should be written to the transmit FIFO.
4	tLteTrigger	Transmit count <= trigger When set, the tLteTrigger field indicates that the transmit FIFO has dropped below or equal to the value specified in the TFFTRG trigger-level register. This is used to let software know that more data should be written to the transmit FIFO. To clear this interrupt, enough data must be written to the transmit FIFO so that the current FIFO count is greater than the trigger.

Table 5-227 SUCGPU_ ISR/ISR_R, SUCSCO_ ISR/ISR_R Bit Definitions (0x0400_A200, 0x0400_AA00 and 0x0400_A204, 0x0400_AA04) (Continued)

Bits	Symbol	Meaning
3	rsm_timeout	Receive character time-out When set, this field indicates that the receive FIFO is not empty and no new character has been received for a time equivalent to receiving 48 bits at the current baud rate. This enables software to process the current entries in the receive FIFO, even though the trigger has not been reached. To clear this interrupt, all entries in the receive FIFO must be read.
2	rGteTrigger	Receive count >= trigger When set, this field indicates that the receive FIFO has more pieces of data than the value set in the RFFTRG registers. To clear this interrupt, the FIFO must be read until the number of data elements it contains is less than the value of the trigger.
1	almost- GonnaReset	Almost Gonna Reset When set, this field indicates that SYS_PWROK has been deasserted and that the system will lose power within 1-2ms. This interrupt is not clearable. Software will keep executing until power is lost.
/0 领域	RESERVED	Reserved for future use

5.2.15.6.2 SUCGPU_IER/_S _C, SUCSCO_IER/_S/ _C Registers

These are the interrupt enable registers. They mask pending interrupts from signalling the CPU. If a bit in the IER register is cleared, the corresponding interrupt in the ISR will be masked. If a bit is set, and the corresponding bit is set in the ISR, an interrupt will be issued to the CPU.

The IER register is a read/writable register that contains the current state of the interrupt enables. If the Interrupt Enable Set register (IER_S) is written, any bit that is a '1' sets the corresponding bit in the IER register to '1.' If the Interrupt Enable Clear register (IER_C) is written, any bit that is a '1' clears the corresponding bit in the IER. If read, IER_S and IER_C registers return the current state of the IER.

The bits of the IER register correspond exactly to the bit definitions of the Interrupt Status register (ISR). See the ISR register description for more details on the particular interrupts.

Table 5-228 SUCGPU_IER/_S/_C, SUCSCO_IER/_S/_C Bit Definitions (0x0400_A210, 0x0400_A208, 0x0400_A20C and 0x0400_AA10, 0x0400_AA08, 0x0400_AA0C)

Bits	Symbol	Meaning
31:8	RESERVED	Reserved for future use.
7	gpioInterrupt	GPIO interrupt
6	tParityErrDetect	Transmit Parity Error
5	transmitterEmpty	Transmit shift register empty
4	tLteTrigger	Transmit count <= trigger
3	rsm_timeout	Receive character time-out
2	rGteTrigger	Receive count >= trigger
1	almostGonnaReset	Almost Gonna Reset
0	RESERVED	Reserved for future use.

5.2.15.7 Smart Card Deactivation Registers

5.2.15.7.1 SUCGPU_SDSMCR/SUCSC0_SDSMCR Registers.

These registers control the Smart Card deactivation state machine.

Table 5-229 SUCGPU_SDSMCR/SUCSC0_SDSMCR Bit Definitions (0x0400_A280, 0x0400_AA80)

Bits	Symbol	Meaning
31:2	RESERVED	Reserved for future use
1	auto Deactivate- ModeAuto	Auto Deactivate Smartcard When the autoDeactivateMode is set, the Smart Card will be deactivated using a hardware state machine whenever the SMC_INSERT_N sig- nal is deasserted (indicating that the card is being removed or that the over-current sensing mecha- nism was triggered). This has the same effect that the drive Smart Card configuration resistor has, with the exception that software still has control over the Smart Card I/O when the Smart Card is deactivated.
0	deactivate Smartcard	Deactivate the Smart Card When the deactivateSmartcard bit is written with a '1,' it initiates the deactivation of the Smart Card through a hardware state machine. This bit is self-clearing. It clears when the deactivation process has finished. Note that this bit does not in- dicate whether the card is currently deactivated, it just indicates whether the deactivation state ma- chine is idle or not. Writing '0' to this bit has no ef- fect. State after reset is 0x0.

5.2.15.7.2 SUCGPU_SDSMSTATE/SUCSCO_SDSMSTATE Registers

These registers contain the state of the Smart Card deactivation state machine.

Table 5-230 SUCGPU_SDSMSTATE/SUCSCO_SDSMSTATE Bit Definitions (0x0400_A284, 0x0400_AA84)

Bits	Symbol	Meaning
31:3	RESERVED	Reserved for future use.
2:0	sdsmState	Smart Card Deactivate State Machine State. This field is a read-only indicator of the current state of the Smart Card deactivation state machine. The states are encoded as follows: SDSM_SWCONTROL 0b000 SDSM_STARTDEACT 0b001 SDSM_ASSERTRESET 0b010 SDSM_STOPCLK 0b011 SDSM_DATAHIZ 0b100 SDSM_PENOFF 0b101 SDSM_PENOFF 0b101 SDSM_DEACTIVATED 0b110

5.2.15.8 GPIO Registers

The GPIO registers are used to control the input and output characteristics of the sucUnit external signals. Each bit of these registers controls one external signal. The bit encoding that maps the bit position to the signal is identical throughout these registers.

Since these same registers are used to control either UART or Smart Card signals, a common name was chosen to refer to the bits. The following abbreviations are used:

SCDRXD	SMC_DATA or UART_RXD
SCCTXD	SMC_CLK or UART_TXD
SCCDTR_N	SMC_PEN_N or UART_DTR_N
SCCRTS_N	SMC_RESET_N or UART_RTS_N
SCCCTS_N	SMC_INSERT_N or UART_CTS_N
SCCDCD_N	UART_DCD_N

5.2.15.8.1 SUCGPU_IOOD/SUCSCO_IOOD Registers

These registers provide the output type control. If a bit is a '1,' the corresponding signal will be an open-drain signal when configured as an output. If a bit is '0,' the corresponding signal will be an totem-pole signal when configured as an output. An external pullup is required in order to use the open-drain mode.

Table 5-231 SUCGPU_IOOD/SUCSC0_IOOD Bit Definitions (0x0400_A288, 0x0400_AA88)

Bits	Symbol	Meaning
31:7	RESERVED	Reserved for future use.
5	ioOpenDrainDCD_N	UART_DCD_N Open Drain
4	ioOpenDrainSCICTS_N	SMC_INSERT_N/UART_CTS_N Open Drain
3	ioOpenDrainSCRRTS_N	SMC_RESET_N/UART_RTS_N Open Drain
2	ioOpenDrainSCPDTR_N	SMC_PEN_N/UART_DTR_N Open Drain
1	ioOpenDrainSCCTXD	SMC_CLK/UART_TXD Open Drain
0	ioOpenDrainSCDRXD	SMC_DATA/UART_RXD Open Drain

5.2.15.8.2 SUCGPU_SPIOCR/SUCSC0_SPIOCR Registers

These registers control the output functionality of signals when the special-purpose output mode is enabled. The special-purpose mode is enabled by writing a '1' to the corresponding bit of the SPIOEN register. These bits have no effect if the corresponding bit in the SPIOEN register is '0.'

Table 5-232 SUCGPU_SPIOCR/SUCSCO_SPIOCR Bit Definitions (0x0400_A28C, 0x0400_AA8C)

Bits	Symbol	Meaning
31:3	RESERVED	Reserved for future use
1	spioSelectSCCTXD	Drive Smart Card Master Clock If spioSelectSCCTXD is '1,' the Smart Card master clock is driven on the output signal. If set to '0,' the transmit data from the transmit shift register is driven on the output signal. This field has no effect if SPIOEN[1] is '0.' State after reset is 0x0.
0	spioSelectSCDRXD	Drive Receive Parity Error. If spioSelectSCDRXD is '1,' the receive parity error will be indicated on the output by driving the signal low during the Stop bit of the receive data. This should only be used when receiving data from a Smart Card. If spioSelectSCDRXD is '0,' transmit data will be driven on the signal. This should only be used for transmitting data to a Smart Card. This field has no effect if SPIOEN[0] is '0.' State after reset is 0x0.

5.2.15.8.3 SUCGPU_SPIOEN/_C/_S, SUCSCO_SPIOEN/_C/_S Registers

These registers enable a specific purpose signal onto the corresponding output signal. When a bit is '0,' the output will be in general-purpose I/O mode. Software controls the state of the signal using the GPIOVAL and GPIODIR registers. When a bit is '1,' the corresponding bit will be driven directly by the hardware with a specific function. When a bit is '1,' the corresponding bit in the GPIODIR register will be ignored.

Table 5-233 SUCGPU_ SPIOEN/_C/_S, SUCSCO_ SPIOEN/_C/_S Bit Definitions (0x0400_A298, 0x0400_A294, 0x0400_A290, 0x0400_AA98, 0x0400_AA94, 0x0400_AA90)

Bits	Symbol	Meaning
31:6	RESERVED	Reserved for future use
5	Zero	Always reads 0x0
4	Zero	Always reads 0x0
3	spioEnableSCRRTS_N	SMC_RESET_N/UART_RTS_N GPIO Enable When this bit is '1,' the signal drives a hardware-controlled RTS_N (request to send) signal. This signal is used to implement hardware flow control. The signal will be asserted low when the receive FIFO has fewer entries than the trigger level specified by the RFFTRG register. The signal is deasserted (high) when the number of FIFO entries is greater than, or equal to, the trigger level. When spioEnableSCRRTS_N is '0,' the signal is a GPIO.
2	Zero	Always reads 0x0
1	spioEnableSCCTXD	SMC_CLK/UART_TXD GPIO Enable When spioEnableSCDRXD is '1,' the signal will drive either the Smart Card master clock or transmit data, depending on the state of the spioSelectSCCTXD (SPIOCR[1]) bit (see the SUCGPU_SPIOCR/SUCSCO_SPIOCR regis- ter bit definitions). When '0,' the signal will be a GPIO.
0	spioEnableSCDRXD	SMC_DATA/UART_RXD GPIO Enable When this bit is a '1,' the signal drives either transmit data to the Smart Card or drives re- ceive parity error status, depending on the state of the spioSelectSCDRXD (SPIOCR[0]) bit. When '0,' the signal will be a GPIO.

5.2.15.8.4 SUCGPU_GPIODIR_C/_S, SUCSCO_GPIODIR_C/_S Registers

These registers control the direction of the I/O signals when in GPIO mode, as indicated by the SPIOEN register. If a bit in the GPIODIR register is a '1,' the corresponding signal will be an output. If the bit is a '0,' the corresponding signal will be an input.

The GPIODIR registers are readable and writable. The state of the GPIODIR registers is also readable through GPIODIR_S and GPIODIR_C.

The GPIODIR_S register is the set register for GPIODIR. Any bit written as a '1' to GPIODIR_S will set the corresponding bit in GPIODIR. Any zeros written to this register are ignored.

The GPIODIR_C register is the clear register for GPIODIR. Any bit written as a '1' to GPIODIR_S clears the corresponding bit in GPIODIR. Any zeros written to this register are ignored.

Table 5-234 SUCGPU_ GPIODIR/_C/_S, SUCSCO_ GPIODIR/_C/_S Bit Definitions (0x0400_A2A8, 0x0400_A2A4, 0x0400_A2A0, 0x0400_AAA8, 0x0400_AAA4, 0x0400_AAA0)

Bits	Symbol	Meaning
31:6	RESBRVED	Reserved for future use
5	gpioDirDCD_N	UART_DCD_N GPIO Direction
4	gpioDirSCICTS_N	SMC_INSERT_N/UART_CTS_N GPIO Direction
3	gpioDirSCRRTS_N	SMC_RESET_N/UART_RTS_N GPIO Direction
2	gpioDirSCPDTR_N	SMC_PDIR_N/UART_DTR_N GPIO Direction
1	gpioDirSCCTXD	SMC_CLK/UART_TXD GPIO Direction
0	gpioDirSCDRXD	SMC_DATA/UART_RXD GPIO Direction

5.2.15.8.5 SUCGPU_GPIOVAL/_C, SUCSCO_GPIOVAL_C /_S Registers

These registers control the direction of the I/O signals, when in GPIO mode, as indicated by the SPIOEN register. If a bit in the GPIOVAL register is a '1' and the corresponding bit in the GPIODIR register is also a '1' (indicating it is an output), the corresponding signal is set to '1.' If the bit is '0', the corresponding output is set to '0.'

The GPIOVAL register is readable and writable. The state of the GPIOVAL register is also readable through GPIOVAL_S and GPIOVAL_C. The GPIOVAL_S register is the set register for GPIOVAL. Any bit written as a '1' to GPIOVAL_S sets the corresponding bit in GPIOVAL. Any '0's written to this register are ignored. The GPIOVAL_C register is the clear register for GPIOVAL. Any bit written as a '1' to GPIOVAL_S clears the corresponding bit in GPIOVAL. Any '0's written to this register are ignored.

Table 5-235 SUCGPU_GPIOVAL/_C/_S, SUCSCO_GPIOVAL/_C/_S Bit Definitions (0x0400_A2B8, 0x0400_A2B4, 0x0400_A2B0, 0x0400_AAB8, 0x0400_AAB4, 0x0400_AAB0)

Bits	Symbol	Meaning
31:6	RESERVED	Reserved for future use:
5	gpioValDCD_N	UART_DCD_N GPIO Value
4	gpioValSCICTS_N	SMC_INSERT_N/UART_CTS_N GPIO Value
3	gpioValSCRRTS_N	SMC_RESET_N/UART_RTS_N GPIO Value
2	gpioValSCPDTR_N	SMC_PEN_N/UART_DTR_N GPIO Value
1	gpioValSCCTXD	SMC_CLK/UART_TXD GPIO Value
0	gpioValSCDRXD	SMC_DATA/UART_RXD GPIO Value

5.2.15.8.6 SUCGPU_GPIORIER/_C /_S, SUCSC0_GPIORIER/_C/_S, SUCGPU_GPIOFIER/_C /_S, SUCSC0_GPIOFIER/_C/_S Registers

These registers are the rise and fall interrupt enables for the I/O signals. These interrupts detect rising and falling transitions on the inputs, and generate an interrupt to the CPU. The interrupts may be enabled regardless of whether or not the SPIOCR register indicates that the signal is a GPIO.

If any GPIO interrupt is pending and enabled, the GPIOISR register's gpioInterrupt bit (ISR[7]) is set (refer to Table 5-227). If the IER register's gpioInterrupt bit (IER[7]) is enabled (refer to Table 5-228), the sucUnit will signal an interrupt to the busUnit. The busUnit then interrupts the CPU, if the appropriate sucUnit interrupt is enabled in the busUnit.

If a bit in the GPIORIER register is a '1,' rising transitions on the corresponding inputs will update the GPIOISR interrupt status register and generate an interrupt. If the GPIORIER register's bit is zero, the interrupt will be masked and the status register (GPIOISR) will not be updated.

If a bit in the GPIOFIER register is '1,' rising transitions on the corresponding inputs will update the GPIOISR interrupt status register and generate an interrupt. If the GPIOFIER register's bit is zero, the interrupt will be masked and the status register (GPIOISR) will not be updated.

Writing '1's to bits in the GPIORIER_S/GPIOFIER_S registers will set the corresponding bits of the GPIORIER/GPIOFIER registers to '1.' Writing '1's to bits in the GPIORIER_C/GPIOFIER_C registers will set the corresponding bits of the GPIORIER/GPIOFIER registers to zero.

The GPIORIER/GPIOFIER registers mask out the GPIO interrupt status when the GPIORISR and GPIOFISR registers are read. Only the GPIORIER/GPIOFIER register bits that are zero will be masked.

Table 5-236 SUCGPU_ GPIORIER/_C/_S, SUCSCO_ GPIORIER/_C/_S, SUCGPU_ GPIOFIER/_C/_S, SUCSCO_ GPIOFIER/_C/_S Bit Definitions (0x0400_A2C8, 0x0400_A2C4, 0x0400_A2C0, 0x0400_AAC8, 0x0400_AAC4, 0x0400_AAC0, 0x0400_A2D8, 0x0400_A2D4, 0x0400_A2D0, 0x0400_AAD8, 0x0400_AAD4, 0x0400_AAD0)

Bits	Symbol	Meaning
31:6	RESERVED	Reserved for future use
5	gpiolerDCD_N	UART_DCD_N GPIO Interrupt Enable
4	gpiolerSCICTS_N	SMC_INSERT_N/UART_CTS_N GPIO Interrupt Enables
3	gpiolerSCRRTS_N	SMC_RESET_N/UART_RTS_N GPIO Interrupt Enable
2	gpiolerSCPDTR_N	SMC_PIER_N/UART_DTR_N GPIO Interrupt Enable
1	gpioIerSCCTXD	SMC_CLK/UART_TXD GPIO Interrupt Enable
0	gpiolerSCDRXD	SMC_DATA/UART_RXD GPIO Interrupt Enable

5.2.15.8.7 SUCGPU_GPIOISR/_C/_S, SUCGPU_GPIORISR, SUCGPU_GPIOFISR, SUCSC0_GPIOISR/_C/_S, SUCSC0_GPIORISR, SUCSC0_GPIOFISR Registers

These registers are the GPIO interrupt status registers. They record rising transitions, if the corresponding bit is set in the GPIORIER register, and falling transitions if the corresponding bit is set in the GPIOFIER register.

There is only one physical GPIOISR register, but it records both the rise and fall events. If a bit is set in the GPIOISR register, and the corresponding bit is set in either the GPIORIER or GPIOFIER registers, an interrupt will be generated.

When reading the GPIORISR register, all bits that are zero in the GPIORIER will be read as zero. All bits that are set to '1' in the GPIORIER register will read the value in the GPIOISR register.

When reading the GPIOFISR register, all bits that are zero in the GPIOFIER register will be read as zero. All bits that are set to '1' in the GPIOFIER register will read the value in the GPIOISR register.

GPIOISR_C is the clear register for the GPIOISR register. Any '1's written to GPIOISR_C will clear the corresponding bit in the GPIOISR register. This is the method that should be used to clear GPIO interrupts.

GPIOISR_S is the set register for the GPIOISR register. Any '1's written to GPIOISR_S will set the corresponding bit in the GPIOISR register. This should not be used in normal operation, it should only be used for testing interrupts.

Table 5-237 SUCGPU_ GPIOISR/_C/_S, SUCGPU_ GPIORISR, SUCGPU_ GPIOFISR, SUCSCO_ GPIOISR/_C/_S, SUCSCO_ GPIORISR, SUCSCO_ GPIOFISR Bit Definitions (0x0400_A2E8, 0x0400_A2E4, 0x0400_A2E0, 0x0400_A2EC, 0x0400_A2F0, 0x0400_AAE8, 0x0400_AAE4, 0x0400_AAE0), 0x0400_AAF0

Bits	Symbol	Meaning
31:6	RESERVED	Reserved for future use.
5	gpioIsrDCD_N	UART_DCD_N GPIO Interrupt Status
4	gpioIsrSCICTS_N	SMC_INSERT_N/UART_CTS_N GPIO Interrupt Status
3	gpioIsrSCRRTS_N	SMC_RESET_N/UART_RTS_N GPIO Interrupt Status
2	gpioIsrSCPDTR_N	SMC_PEN_N/UART_DTR_N GPIO Interrupt Status
1	gpioIsrSCCTXD	SMC_CLK/UART_TXD GPIO Interrupt Status
0	gpioIsrSCDRXD	SMC_DATA/UART_RXD GPIO Interrupt Status

5.2.15.9 Diagnostic Registers

5.2.15.9.1 SUCGPU_DIAGMODE /SUCSCO_DIAGMODE Registers.

These registers contain the GPU/SC0 diagnostic mode.

Table 5-238 SUCGPU_DIAGMODE/SUCSCO_DIAGMODE Bit Definitions (0x0400_A300, 0x0400_AB00)

Bits	Symbol	Meaning
31:5	RESERVED	Reserved for future use.
4	diagLoopBack	Enable Loop Back UART Mode. Loop-back mode for UART testing. Effectively makes the following connections: UART_TXD driven internally to UART_RXD UART_RTS_N driven internally to UART_CTS_N UART_DTR_N driven internally to UART_DCD_N Loop-back should not be used in Smart Card mode. Loop-back is forced to zero when driving the Smart Card directly.
3.2	RESERVED	Reserved for future use
1:0	diagBusSel	Diag Bus Select. 0b00 Drive zero on the diag bus

5.2.15.10 Choosing Smart Card Clock Divisors

When choosing divisors for the Smart Card master clock and sample clock, a specific relationship between the two clocks must be maintained. The EMV Smart Card specification puts some restrictions on the possible relationships, but there are still many configurations. This section shows you how to determine the proper divisors for EMV-compliant Smart Cards.

The following relationships must be met according to the EMV spec:

Where:

- f(SMC_CLK) is the frequency of the Smart Card master clock.
- f(sampClk) is the frequency of the Smart Card sample clock (which is defined to be 16 times the bit rate)
- effBR is the effective baud rate
- D is the clock multiplier (which is restricted to be between 16 and 1/16 according to the EMV spec)
- F is the clock divider (which is always exactly 372 according to the EMV spec)

Also, according to the definition of the mastClkDiv and sampClkDiv registers:

$$f(SMC_CLK) = \frac{f(SYS_2XCLKIN)}{(mastClkDiv + 1) * 2}$$

$$f(sampClk) = \frac{f(SYS_2XCLKIN)}{(sampClkDiv + 1)}$$

mastClkDiv and sampClkDiv should be chosen to be exact integers to provide the proper relationship.

So, we solve for sampClkDiv. Then we choose mastClkDiv so that it gives us a Smart Card clock between 1MHz and 5MHz, and an exact integer for sampClkDiv.

The following equations summarize the requirements for choosing proper divisors:

$$f(SMC_CLK) = \frac{f(SYS_2XCLKIN)}{(mastClkDiv + 1) * 2}$$

mastClkDiv and sampClkDiv must be exact integers

Here are some examples for various values of D, assuming that f(SYS_2XCLKIN) is 83MHz and F=372, as specified by the EMV spec:

- at D = 16: mastClkDiv=31, sampClkDiv=92, f(SMC_CLK) = 1.29 MHz (mastClkDiv+1 must be multiple of 32 for sampClkDiv to be an integer)
- at D = 8: mastClkDiv=15, sampClkDiv=92, f(SMC_CLK) = 2.59 MHz
 (mastClkDiv+1 must be multiple of 16 for sampClkDiv to be an integer)
- at D = 4: mastClkDiv=15, sampClkDiv=185, f(SMC_CLK) = 2.59 MHz (mastClkDiv+1 must be multiple of 8 for sampClkDiv to be an integer)
- at D = 2: mastClkDiv=11, sampClkDiv=278, f(SMC_CLK) = 3.46 MHz (mastClkDiv+1 must be multiple of 4 for sampClkDiv to be an integer)
- at D = 1: mastClkDiv=11, sampClkDiv=557, f(SMC_CLK) = 3.46 MHz (mastClkDiv+1 must be multiple of 2 for sampClkDiv to be an integer)
- at D = 1/2: mastClkDiv=10, sampClkDiv=1022, f(SMC_CLK) = 3.77 MHz (mastClkDiv may be any value)
- at D = 1/4: mastClkDiv=10, sampClkDiv=2045, f(SMC_CLK) = 3.77 MHz (mastClkDiv may be any value)
- at D = 1/16: mastClkDiv=10, sampClkDiv=8183, f(SMC_CLK) = 3.77 MHz (mastClkDiv may be any value)

When possible, mastClkDiv should be chosen so that it produces a master Smart Card clock between 3.2MHz and 4.0MHz.

The following is from Gemplus Card International, "How to Develop and/or Use Smart Card Readers/Writers", p10-11 (Section 2.3, Temporal Characteristics of the Signals):

"Note: In banking, it is normal to have fi = 3.571200 MHz as it allows a direct transfer rate of 9600 bits/s on the I/O line when the F parameter equals 372, the default value during "Answer to Reset."

It is recommended to operate all interfaces with 3.2MHz <= fi <= 4MHz rather than in the range [1MHz, 5MHz] specified by the Standard ISO-7816-3, so that:

- the maximum frequencies of certain cards are not exceeded
- the temporization loops are not inconsiderately shortened or lengthened which could be used, for example, by the code of the card's micro-chip to generate programming pulses for the EEPROM memory

 the frequency does not go below the frequency detection threshold of certain components of the card"

5.3 Programming the Graphics Engine

5.3.1 Graphics Display Modes

5.3.1.1 SOLO1 Mode

When the SOLO1 ASIC is powered up, it is in the normal *Frame Buffer Mode* used by FIDO1. In this mode, the software is responsible for maintaining a frame buffer in memory, and the vidUnit 's DMA engine reads and displays that frame buffer (see Figure 5-10).

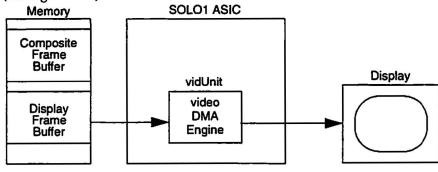


Figure 5-10 Frame Buffer Mode

5.3.1.2 GRFX Direct Mode (Ping-Pong Mode)

When the SOLO_GRFX bit is enabled and the write-back is disabled, we are in GRFX Direct Mode. In this mode, the gfxUnit assembles a video image on-the-fly, keeping pace with the scanning beam of the TV. This is possible because NTSC and PAL monitors scan at a particularly slow rate. Although this certainly introduces limitations on the number of objects that can be assembled on a given scanline, it is always possible to display an entire full-screen frame buffer, allowing image assembly like a conventional graphics system.

Since the data is sourced from the memory at a much higher rate than the data must be scanned out to refresh an NTSC or PAL TV, it is possible to load several layers of graphics in the Scanline buffer during a single NTSC or PAL scanline time. Thus, by providing two scanline buffers in SOLO1, it is possible to simultaneously load graphics data into one Scanline buffer, while refreshing the TV by reading from the other Scanline buffer. When it is time to scan out the next scanline to the TV, SOLO1 swaps the buffers, scanning out the one that was just loaded to the TV while commencing to load the one that was just used for scanning out. SOLO1 continues this "ping-ponging" between Scanline buffers until the entire video field has been refreshed, and it continues to do so for every field thereafter.

In this mode, software maintains the display list (yMap entries), celRecord structure, and the necessary textures and codebooks in memory. The gfxUnit will compose the picture line-by-line and send it to the video encoder for display.

For GRFX Direct Mode, each line of the display must be composed by the hardware in one video line time. While a given line of video is being read from one Scanline buffer and sent to the display, the next line is being composed into the second Scanline buffer.

It is possible that an image could be presented to the gfxUnit that is too complex to process in one scanline time. If the end of a line is reached before compositing is complete, then the current line is aborted, an interrupt is generated, and processing moves on to the next line. The line that was aborted (with all of the cels that have been processed so far) is displayed. The cels toward the end of the yMap, that weren't processed, are not displayed.

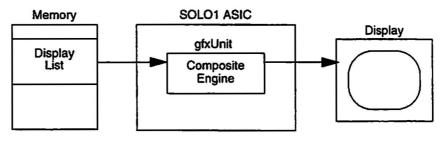


Figure 5-11 GRFX Direct Mode

5.3.1.3 GRFX Write-back Mode

For GRFX Write-back Mode, both the gfxEn and WriteBack bits, from the GRFX_CONTROL register, are enabled. In this mode, the gfxUnit is responsible for composing the display image and writing the image back to a full frame buffer in memory. The video DMA engine in the vidUnit then reads the data in the frame buffer and sends it on to the display device.

To perform a write-back, SOLO1 composites into one line buffer, and when that line is complete, the line is written back into a full frame buffer stored in SGRAM. The "spot mode" video DMA is then responsible for displaying the frame buffer on the screen. Writing back and reading out again takes up more memory bandwidth, but in this mode, the compositing operation is no longer bounded by the video line time.

This mode is useful for very complex images where the compositing time for one line of the image is longer than the video display line time (see Figure 5-12).

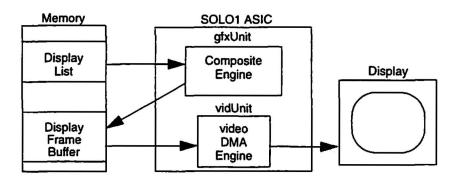


Figure 5-12 GRFX Write-back Mode

There are two Write-back modes: *Field* mode and *Frame* mode. Field mode has an update rate of 60Hz, because each field of write-back is enabled separately. Frame mode has a 30Hz update rate, because each frame is completely written back before the next frame can be updated.

5.3.2 Programmable Features

5.3.2.1 Blending

The compositing operation is a read-modify-write operation. The Cel being written is blended into the Scanline buffer, by reading each pixel value already in the buffer, calculating a weighted average of the new and the old values (based on the Alpha values), and writing the result back into the Scanline buffer.

Alpha is an 8-bit unsigned value where A = 0 causes the foreground to be transparent, A = 255 causes the foreground to be opaque, and intermediate values cause the foreground to be blended into the background. The full range of Alpha [0, 255] is utilized by SOLO1 and all values are valid.

For this blend operation, the background pixel is the pixel value already in the Scanline buffer. The foreground pixel is the new pixel value being written for the current Cel.

Note: In both 4:4:4 and 4:2:2 formats, if Y = 0xff, the foreground Alpha is forced to 0 (fully transparent)

5.3.3 Texture Formats

5.3.3.1 Pixel Formats

SOLO1 supports only two pixel formats. The first format uses 32 bits for each pixel, which includes an 8-bit foreground Alpha value. The second format uses 16 bits for each pixel, by sharing the chrominance values between two adjacent pixels, and has no per-pixel Alpha value.

The pixels in the SOLO1 architecture are always in YCbCr color space. Y is the luminance value and describes the brightness of a pixel. Its range is between 0 and 219. Cb and Cr are the chrominance values and together they describe the color of a pixel. The Cb and Cr values range from -112 to 112.

If your data is in the range Y [16, 235] and Cr, Cb [16, 240], see the Direct 4:2:2 offset mode in Table 5-239.

The software performs the conversion from RGB color space according to the following format (assuming R, G, and B range from 0 to 255):

```
Y = 0.257R + 0.504G + 0.098B

Cr = 0.439R - 0.368G - 0.071B

Cb = -0.148R - 0.291G + 0.439B
```

5.3.3.1.1 Gamma Correction: SOLO1 performs no gamma correction. Gamma correction, or no gamma correction, is the responsibility of the system software, although SOLO1 can often implement gamma correction with a properly biased Codebook in one of the VQ or color lookup modes.

5.3.3.1.2 4:4:4 Pixel Format This pixel format is described as 4:4:4 because for every four luminance values (Y), there are also four Cb values and four Cr values. This format includes an 8-bit foreground Alpha value for each pixel (see Figure 5-13).

Υ	Cb	Α	Cr	
32			0	ĺ

Figure 5-13 4:4;4 Pixel Format

5.3.3.1.3 4:2:2 Pixel Format This pixel format is described as 4:2:2 because for every four luminance values (Y), there are two Cb values and two Cr values. In other words, pixels 0 and 1 use the same chrominance information. This format is also referred to as YCbYCr format and is equivalent to 16 bit/pixel. Each pair of adjacent Y pixels share a common Cb and Cr, resulting in less memory usage than there would be if each Y had its own Cb/Cr values.

There is not a full Alpha stored per pixel in 4:2:2 format, so Alpha blending is limited to a global Alpha for the entire Cel (see Figure 5-14).

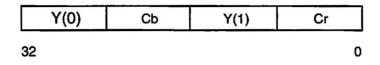


Figure 5-14 4:2:2 Pixel Format

5.3.3.2 Direct vs. VQ Textures

For *Direct* textures, all of the pixels comprising an image are stored sequentially in memory. SOLO1 reads the pixels, performs the compositing operation and displays the resulting image.

SOLO1 also supports a form of *Vector Quantization* (VQ), which is a simple, yet high-quality compression/decompression method for images. VQ textures are composed of vectors instead of pixels. Each vector points to a pixel in the Codebook associated with that texture. SOLO1 first reads a group of vectors (16), then reads the corresponding Codebook entries. Both types of textures are illustrated in Figure 5-15.

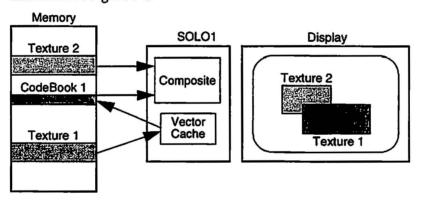


Figure 5-15 Direct vs. VQ Textures

The VQ8 Format stores eight bits per vector. This corresponds to 256 possible entries in the associated Codebook.

The **VQ4 Format** stores four bits per vector, corresponding to 16 possible entries in the associated Codebook.

Codebook entries can be in either 4:4:4 pixel format or 4:2:2 format. A Codebook entry in 4:4:4 pixel format is a 32-bit word containing a Y, Cr, Cb, and Alpha value for each pixel. If 4:2:2 pixel format is used, each Codebook entry contains information for four pixels and takes up two 32-bit words. The horizontal and

vertical position of the desired pixel is used to determine which pixel color is used out of the four possible pixels stored at that entry. The various texture formats are listed in Table 5-239.

Table 5-239 Texture Format Summary

Mode	Texture	codeBook	Pixel format	
Direct 4:4:4	32 bits per pixel	none	ҮСЬАСт	
Direct 4:2:2	16 bits per pixel	none	YCbYCr	
Direct 4:2:2 Offset	Same as Direct 4:2:2, except that range of Y is [16, 235] instead of [0, 219], and range of Cr/Cb is [16, 240] instead of [-112, 112].			
VQ8 4:4:4	8 bits per vector	256 entry x 32 bits per entry	YCbACr	
VQ8 4:2:2	8 bits per vector	256 entry x 16 bits/pix x 4 pix	YCbYCr	
VQ4 4:4:4	4 bits per vector	16 entry x 32 bits per entry	YCbACr	

5.3.4 Codebook Formats

Codebooks are arrays of 32-bit pixel specifications. Codebooks must not cross page boundaries or else there will be significant performance penalties. Also, Codebooks can not be read directly from ROM.

Codebooks for 8-bit vectors are 256 entries long. They may be shorter than 256 entries if it is known that the vectors do not utilize all 256 entries.

Codebooks for 4-bit vectors are 16 entries long. As with 8-bit Codebooks, they can be shortened.

5.3.4.1 On-chip Codebook

The gfxUnit contains a 16-entry on-chip Codebook that may be used in lieu of a Codebook in RAM. The gfxUnit is able to read from this Codebook while simultaneously reading a vector from memory, substantially improving its compositing throughput.

The on-chip Codebook is selected whenever the codebookBase parameter in a CelRecord is 0. MicroCelRecords always utilize the on-chip Codebook.

The on-chip Codebook only contains 16 entries and, as so can only work with 4:4:4 Pixel Format and 4-bit vectors. Attempting to use the on-chip Codebook in any other mode will yield unpredictable results.

The on-chip Codebook is loaded by specifying a CelRecord as a Codebook Load CelRecord (see the celMode Byte section). When the gfxUnit encounters a Codebook Load CelRecord, it loads the Codebook data, pointed to by the CelRecord, into the on-chip Codebook. This Codebook remains on-chip until another is loaded over it.

5.3.5 Data Structures

The client software creates yMap and CelRecord structures that tell SOLO1 how to construct the final image for display. The display image is divided up into layers, with each layer having its own set of data structures.

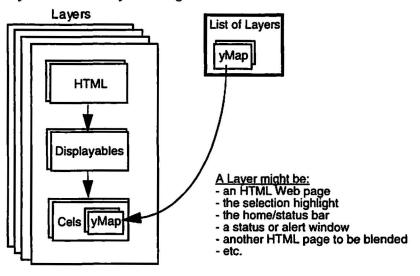


Figure 5-16 Data Structures

5.3.5.1 yMap Structure

The yMap structure contains the location of the top edge and the heights of all of the Cels in the displayed image. By comparing that information with the current line count, SOLO1 can determine whether a given Cell is active on the current line. The Cel is active if:

```
(yCount < topLine + height) & (yCount >= topLine)
```

The yMap structure is illustrated in Table 5-240.

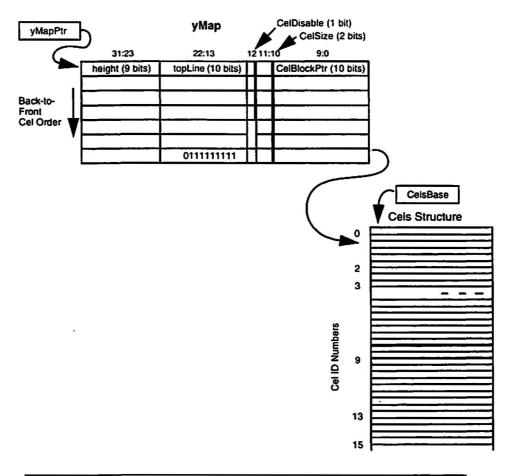


Figure 5-17 yMap Structure

The SOLO1 register yMap_Ptr points to the location of the yMap structure in memory. The yMap structure contains one word for each Cel defined. After reading a one-word entry, SOLO1 determines whether the specified Cell is active for the current scanline. This is done by comparing the topLine parameters with the current y count. If the Cel is active, the appropriate Cel structure is loaded and the pixels for the current scanline are blended into the Scanline buffer. If the topLine parameter equals +511 (0111111111), then the entry is a termination entry. This signifies the end of processing for the current scanline.

The yMap parameters and their definitions are in Table 5-240.

Table 5-240 yMap Structure

Parameter	Size	Definition
height	9 bits unsigned	Number of lines the cel occupies. If the height = 0, that means "always draw."
topLine	10 bits signed	Top line of the cel. If topLine = 511, then this is a termination entry signifying the end of the yMap structure.
CelDisable	1 bit	0 = normal operation 1 = do not draw cel and skip to the next yMap entry
Celsize	2 bits	0 = MicroCelRecord 1 = MiniCelRecord 2 = full CelRecord 3 = reserved
CelBlockPtr	10 bits unsigned	Offset into the Cels Structure. Number of double words from the CelsBase (the 3 LSB's are assumed to be 0).

Note: The CelDisable bit has "priority" over the termination entry.

Table 5-241 CelDisable Bit Definition

Terminated	Disabled	action
0	0	draw cel
0	1	disable drawing: go to next yMap entry
1	0	terminate line
1	1	disable drawing: go to next yMap entry

5.3.5.2 Cels structure

Each Cel is associated with a CelRecord which specifies all parameters of a Cel, except for its y position on the screen and its height. The Cel's structure is a list of all the CelRecords, packed one after another.

CelRecords vary in size from eight bytes to 48 bytes. Cels in the Cels structure are numbered on the basis of double-words. Note that Cel numbers are skipped when a CelRecord is larger than the minimum eight bytes.

A CelBlock is a collection of consecutive Cels that share the same *topLine* and *Height* position on the screen. An example of this is a line of text where each character is a separate Cel, but all characters have the same characteristics with respect to the *y* axis of the screen.

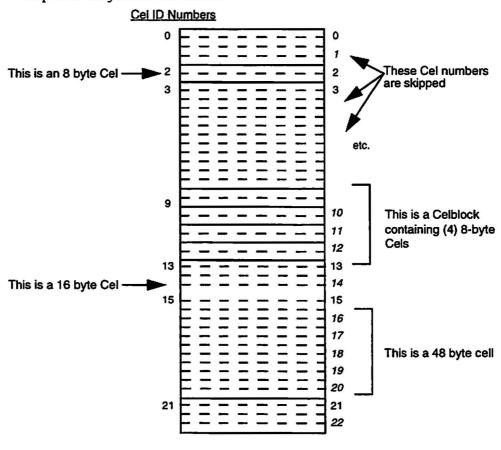


Figure 5-18 Cels Structure Numbering

5.3.6 CelRecords

The CelRecord structure specifies all parameters of a Cel, except for its y position on the screen. There are three types of CelRecords: full CelRecord, miniCelRecord, and microCelRecord. The largest is the full CelRecord, which allows all possible transformations and Cel modes. The smallest is the microCelRecord, which is optimized for text characters. MiniCelRecords support scaling, but do not support rotations or perspective.

The structures are designed so that the microCelRecord is a subset of the miniCelRecord and the miniCelRecord is a subset of the full CelRecord. This simplifies things considerably. The SOLO1 State Machine first loads default values into the CelRecord registers. It then reads in the CelRecord parameters, which overwrite all of the register defaults (if it is a full CelRecord), or a portion of the register defaults (if it is a miniCelRecord or a microCelRecord).

Table 5-242 CelRecord Parameters

Parameter	# bits	Description		
celMode	8 bits (unsigned)	[7]	lastCel 0 = false 1 = true	
		[6]	if celType !=Load Data textureMem 0 = RAM 1=Reserved. Do not use.	if celType = Load Data [6:4]LoadDataType 0 = loadCodeBook
		[5:4]	if celType !=Load Data background Alpha 0 = background Alpha = 1-A 1 = Force background Alpha = 1 2 = Force background Alpha = 0 3 = (1-A) * BlendColorRegister	1 = loadYMapBase 2 = loadCelsBase 3 = reserved 4 = loadInitcolor 5 = loadYMapBase- Master 6 = loadCelsBase- Master 7 = reserved
		[3:1]	celType 0 = VQ8 4:2:2 1 = Dir 4:2:2 2 = VQ8 4:4:4 3 = Dir 4:4:4 4 = Dir 4:2:2 Offset 5 = Dir 4:2:2 A 6 = LoadData 7 = VQ4 4:4:4	
		[0]	WebTV Port 0 = Normal operation 1 = Reserved. Do not use.	

Table 5-242 CelRecord Parameters (Continued)

Parameter	# bits	Description
topOffset	8 bits (unsigned)	The difference between the top of the Cel and the top of the Cel-Block. topLine topOffset topOffset bottomOffset
texture- RowLongs Low	4 bits (unsigned)	The stride of a texture as stored in SGRAM. Value = ((TextureMap stride in u) -1)/4. This is broken up into textureRowLongsHigh (8 bits) and textureRowLongsLow (4 bits). The microCelRecord only requires low bits since the texture width is limited to 64 bytes. Default value = 0x0.
xLeftStart (integer)	10 bits (signed)	The x-coordinate (in screen space) of the upper left-hand corner of the Cel. In the CelRecord, this parameter is split up into 10 bits of integer and 10 bits of fraction (see the xLeftStartFrac entry, later in this table). Default value = 0x0.
bottom- Offset	8 bits (unsigned)	The difference between the bottom of the Cel and the bottom of the CelBlock.
textureBase	24 bits (unsigned)	A pointer to the texture in memory. Value = (TextureMap Base Address)/4. This address points to the pixel where at u, v = (0,0). Since u and v can range from -128 to 127, this may be in the middle of the texture in memory.
texture- RowLongs- High	8 bits (unsigned)	The stride of a texture as stored in SGRAM. Value = ((TextureMap stride in u) -1)/4. This is broken up into textureRowLongsHigh (8 bits) and textureRowLongsLow (4 bits). The microCelRecord only requires low bits since the texture width is limited to 64 bytes. Default value = 0x0.
duxCenter	8 bits (unsigned 4.4)	The "dux" parameter represents the rate of change of the u coordinate with respect to x. It is separated into three parts: dux-Center (middle eight bits), duxMSN (the four bits of the most significant nibble), and duxLSN (the four bits of the least significant nibble). Default value = 0x10.
xRightStart (integer)	10 bits (signed)	The x-coordinate (in screen space) of the upper right-hand corner of the Cel. This parameter is split up into 10 bits of integer and 10 bits of fraction (see the xRightStartFrac entry, later in this table). Default value = 0x0.

Table 5-242 CelRecord Parameters (Continued)

Parameter	# bits	Description
global- Alpha	8 bits (signed)	Alpha blending factor that controls the blending of an entire Cel. Default value = 0xff.
codeBook- Base	24 bits (unsigned)	A pointer to the first word of the codeBook stored in SGRAM. Value = (CodeBook Base Addr)/4. If this parameter is zero, this signifies that the on-chip codeBook is to be used instead of a codeBook in SGRAM. Default value = 0x0.
uStart	8.8 bits (signed)	The horizontal coordinate (in texture space) of the upper left-hand corner (in screen space) of the Cel. Default value = 0x0.
duRow Adjust	16 bits (signed 8.8)	The rate of change of the u coordinate with respect to the left edge of the Cel. Default value = $0x0$.
vStart	16 bits (signed 8.8)	The vertical coordinate (in texture space) of the upper left-hand corner (in screen space) of the Cel. Default value = 0x0.
dvRow Adjust	16 bits (signed 8.8)	The rate of change of the v coordinate with respect to the left edge of the Cel. Default value = $0x0100$.
xLeftStart- Frac (fraction)	10 bits (signed)	The x-coordinate (in screen space) of the upper left-hand corner of the Cel. In the CelRecord, this parameter is split up into 10 bits of integer and 10 bits of fraction (see the xLeftStart entry, earlier in this table). The full CelRecord is the only type of CelRecord that uses the fractional bits. Default value = 0x0.
dxLeft	20 bits (signed 10.10)	This determines the slope of the left edge of the Cel. Default value = 0x0.
xRight- StartFrac (fraction)	10 bits (unsigned)	The x-coordinate (in screen space) of the upper right-hand corner of the Cel. This parameter is split up into 10 bits of integer and 10 bits of fraction (see the xRightStart entry, earlier in this table). The full CelRecord is the only type of CelRecord that uses the fractional bits. For mini and micro cels, the fractional bits are assumed to be zero.
dxRight	20 bits (signed 10.10)	This determines the slope of the right edge of the Cel. Default value = 0x0.
dvx	16 bits (signed 8.8)	The rate of change of the v coordinate with respect to x . Default value = $0x0$.

Table 5-242 CelRecord Parameters (Continued)

	Table 3-242 Cerketoru Furumeters (Continueu)				
Parameter	# bits	Description			
uMask	8 bits (unsigned)	The uMask parameter allows textures to be repeated by causing the values of u to wrap. 0xFF: Normal operation: (repeat after 256) 0x7F: wrap (only using 128 positive u values) 0x3F: wrap (only using u = [0, 63] 0x01: wrap (only using u = 0, u = 1) Default value = 0xff.			
vMask	8 bits (unsigned)	The vMask parameter allows textures to be repeated by causing the values of v to wrap. 0xFF: Normal operation: (repeat after 256) 0x7F: wrap (only using 128 positive v values) 0x3F: wrap (only using $v = [0, 63]$ 0x01: wrap (only using $v = 0$, $v = 1$) Default value = 0xff.			
duxMSN	4 bits (signed)	The rate of change of the u coordinate with respect to x . Default value = $0x0$.			
duxLSN	4 bits (signed)	The rate of change of the u coordinate with respect to x . Default value = $0x0$.			
YBlendCol- or	8 bits	BlendColor for special Alpha mode where this solid color is used instead of the background pixel. [23:16]=Y Default value = 0xDB0000 (White).			
CbBlend- Color	8 bits	BlendColor for special Alpha mode where this solid color is used instead of the background pixel. [15:8]=Cb Default value = 0xDB0000 (White).			
CrBlend- Color	8 bits	BlendColor for special Alpha mode where this solid color is used instead of the background pixel. [7:0]=Cr Default value = 0xDB0000 (White).			

5.3.6.1 Full CelRecord

The CelRecord structure is exactly 48 bytes long, including reserved fields, which must be 0. Note that several parameters are split into pieces throughout the structure. This is done to simplify SOLO1's internal architecture.

The full CelRecord structure contains all of the information necessary to apply spatial transformations upon a Cel.

A CelRecord must be double word (64-bit) aligned and should not cross 2KB page boundaries for optimal performance.

The information supplied by the Full CelRecord is illustrated in Figure 5-19.

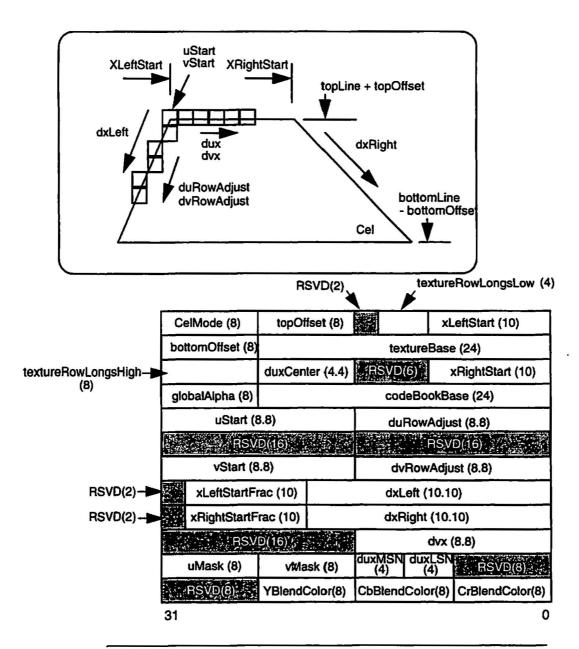


Figure 5-19 Full CelRecord Parameters

5.3.6.2 MiniCelRecord

The MiniCelRecord includes parameters for image scaling, but not for warping or rotations. At only 16 bytes, it saves memory and memory cycles for those Cels that can use it (see Figure 5-20).

Most of the parameters of the CelRecord structure defined above are used for spatial manipulation of images. If all you need to do is display a TextureMap without 3D mapping, you can use the MiniCelRecord structure which requires only 16 bytes of storage.

The MiniCelRecord is identical to the base static parameters of the full CelRecord, except that the "dux" parameter is now used for both dux and dvRowAdjust. Thus, a MiniCelRecord permits image scaling, but only if the scaling factors in u and v are equal.

A MiniCelRecord must be double-word (64-bit) aligned and should not cross 2KB page boundaries for optimal performance.

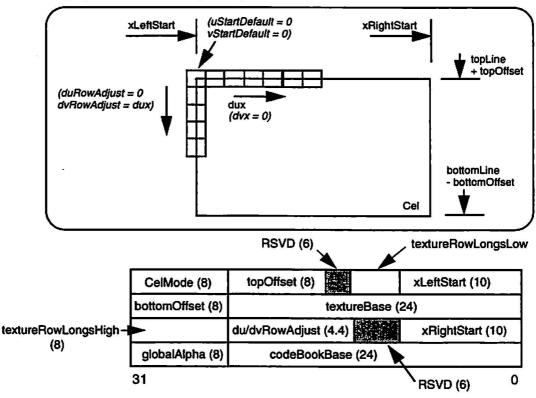


Figure 5-20 MiniCelRecord

Although a MiniCelRecord is missing many of the parameters of a full CelRecord, SOLO1 handles MiniCelRecord Cels exactly as it would full CelRecord Cels. When SOLO1 loads a MiniCelRecord into its internal registers, it uses default values for the parameters that are missing. So SOLO1's internal state is essentially the same upon loading a MiniCelRecord.

The default values used prior to loading a MiniCelRecord are as follows:

```
#define
         uStartDefault0
#define
         vStartDefault0
         duxDefault1
#define
#define
         dvxDefault0
#define
         duRowAdjustDefault 0
#define
         dvRowAdjustDefault 0
#define xLeftStartDefault0
#define xRightStartDefault 0
#define
         dxLeftDefault 0
#define
         dxRightDefault 0
#define
         uMaskDefault 0xff// no u masking
#define
         vMaskDefault 0xff// no v masking
```

5.3.6.3 MicroCelRecord

The MicroCelRecord is targeted for use with text characters, special Cels, or other Cels that use the VQ4 4:4:4 format. It is only 8 bytes in size.

The on-chip Codebook is always used for a microCelRecord. Only the least significant bits of the textureRowLongs parameter are required, since these Cels are limited to 64 bytes in width. Also, xRightStart is omitted since it can be calculated by adding textureRowLongs to xLeftStart.

Even though the MiniCelRecord structure has a modest number of parameters, for some applications (e.g. putting up a text character), there are more parameters than are needed.

SOLO1 has an on-chip 16-entry Codebook (refer to Section 3.4.4.1). If this Codebook is used (assuming we are in VQ mode), then a pointer to an external Codebook is unnecessary. Also, if the width of the displayed image is equal to the full width of the TextureMap, then xRightStart can be calculated by adding textureRowLongs to xLeftStart. Finally, if we are only dealing with simple images, a large textureRowBytes is unnecessary.

Thus, we have the MicroCelRecord, only 8 bytes long:

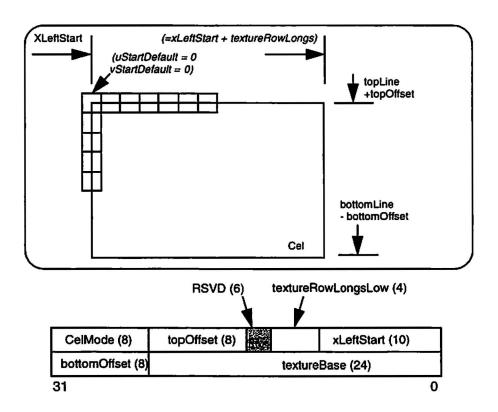


Figure 5-21 MicroCelRecord

A MicroCelRecord must be double word (64-bit) aligned.

As with the MiniCelRecord, SOLO1 deals with the MicroCelRecord by loading registers corresponding to missing parameters with default values. In addition to the MiniCelRecord default values, SOLO1 loads the following MicroCelRecord default values:

```
#define textureRowLongsHighDefault 0
#define CodeBookBaseDefault 0 // use internal Codebook
#define globalAlpha 0xff // fully opaque
And, xRightStart is calculated as follows:
```

xRightStart = xLeftStart + textureRowLongs;

5.3.6.4 LoadData "Special" microCelRecords

Table 5-243 Summary of LoadData microCelRecords

textureMem	background Alpha	type of loadData command
0	00	loadCodeBook
1	00	loadInitColor
0	01	loadYMapBase
1	01	loadYMapBaseMaster
0	10	loadCelsBase
1	10	loadCelsBaseMaster

Load CodeBook MicroCelRecords

SOLO1 decodes the celMode byte in the celRecord to see if the celRecord has a celType that equals "LoadData." If so, then the backgroundAlpha bits and the textureMem bit are decoded to see what type of LoadData microCelRecord it is. If the backgroundAlpha bits are 0x0, and the textureMem bit = 0, then it is a LoadCodeBook command.

The LoadCodeBook command causes SOLO1 to load the on-chip codebook from the RAM address pointed to by textureBase (value = RAM address/4). All other MicroCelRecord parameters are ignored.

Load YMapBase MicroCelRecord

SOLO1 decodes the celMode byte in the celRecord to see if the celRecord has a celType that equals "LoadData". If so, then the backgroundAlpha bits are decoded to see what type of LoadData microCelRecord it is. If the backgroundAlpha bits are 0x1, then it is a LoadYMapBase command. If the textureMem bit = 0 = RAM, then the yMapBase register is loaded. If the textureMem bit = 1 = ROM, then the yMapBaseMaster register is loaded.

The LoadYMapBase command causes SOLO1 to reload the yMapBase register or the yMapBaseMaster register from the address pointed to by textureBase. All other MicroCelRecord parameters are ignored.

Load CelsBase MicroCelRecord

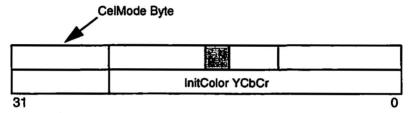
SOLO1 decodes the celMode byte in the celRecord to see if the celRecord has a celType that equals "LoadData1." If so, then the background Alpha bits are decoded to see what type of LoadData microCelRecord it is. If the

backgroundAlpha bits are 0x2, then it is a LoadCelsBase command. If the textureMem bit = 0 = RAM, then the CelsBase register is loaded. If the textureMem bit = 1 = ROM, then the CelsBaseMaster register is loaded.

The LoadCelsBase command causes SOLO1 to reload the CelsBase register from the address pointed to by textureBase. All other MicroCelRecord parameters are ignored.

Load InitColor MicroCelRecords

SOLO1 decodes the celMode byte in the celRecord to see if the celRecord has a celType that equals "LoadData." If so, then the backgroundAlpha bits and the textureMem bit are decoded to see what type of LoadData microCelRecord it is. If the backgroundAlpha bits are 0x0, and the textureMem bit = 1, then it is a LoadInitColor command.



The LoadInitColor command causes SOLO1 to load the InitColor register. The textureBase bits are now used for the new InitColor value. All other MicroCelRecord parameters are ignored.

5.3.6.4.1 Screen Coordinates xLeft, xRight, dxLeft, dxRight, topLine, height, topOffset, bottomOffset Screen coordinates reference the pixel locations on the TV screen. The horizontal and vertical dimensions are represented by x and y, respectively. Both x and y are ten-bit signed numbers extending from -512 to 511.

SOLO1 only operates on integral y values. Thus, a pixel coordinate may only be specified to an accuracy of one scanline. However, SOLO1 does operate on fractional x values (10 bits of fraction). Thus, a pixel can be can be specified in x to within 1/1024th of a pixel, and this address is truncated to the closest integral x location when it is written to the Scanline buffer.

Screen space parameters in x are 10.10 signed fixed point values since we need to span a range of 1024 pixels with sub-pixel positioning.

Note: Screen space parameters in y: topLine, height, topOffset, bottomOffset, and yOffset are 10.0 signed values since we do not need sub-pixel positioning.

The center of the screen is chosen to be x=0, y=0, so that data structures centered on the screen for NTSC will also be centered for PAL and visa versa. If you prepare data for SOLO1 that works for NTSC, it will also work for PAL, although

it will be inset (centered) on the PAL TV. The reverse, however, is not necessarily true since it is possible to specify PAL coordinates which are outside of the scanning area of NTSC.

Cels may be specified to be located anywhere within the 10-bit (x,y) screen coordinate range. Pixels outside of the active display area are not output from SOLO1.

The parameter increases by 1 each frame line, and increases by 2 each field line. Thus, the top and bottom active NTSC lines are y = -239 and y = 240, respectively. The top and bottom active PAL lines are y = -279 and y = 280, respectively. y values above and below these limits are not displayed.

The center pixel horizontally of active video is designated to be x=0. Thus, the left-most and right-most NTSC pixels are x=-320 and x=319, respectively. The left-most and right-most PAL pixels are x=-384 and x=383, respectively. Pixels with x values above and below these limits are not displayed. NTSC and PAL screen coordinates are shown in Figure 5-22.

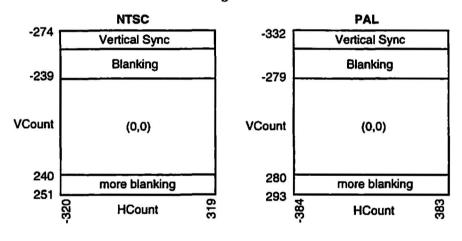


Figure 5-22 Screen Coordinates

5.3.6.4.2 Texture coordinates: uStart, vStart, dux, dvx, duRowAdjust Texture coordinates reference the pixel locations on a TextureMap. The horizontal and vertical texture map coordinates are represented by u and v, respectively.

The u and v parameters are used to index into the Texture Map. They are signed 8.8 fixed-point numbers. The fractional part of u and v is used in the mapping operation, but truncated before the pixel lookup operation.

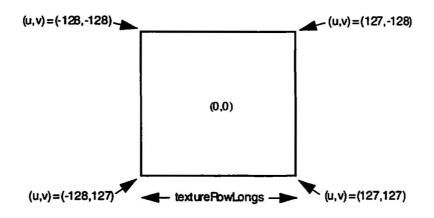


Figure 5-23 Texture Coordinates

TextureMaps are stored in memory in big-endian format with the center pixel stored at address *textureBase*. The data for the TextureMap is stored before and after TextureBase in memory.

5.3.7 gfxUnit Programming Suggestions

5.3.7.1 Using yMap "Subroutines"

One issue with yMaps is that it is possible to construct an image which results in very inefficient yMap processing. For example, consider the case where there are a large number of CelBlocks located near the top of the screen and a large number near the bottom of the screen, but not much overlap in between. In this situation, on a given scanline we will have a large number of yMap entries which are above or below the scanline and therefore do not result in the loading of CelBlocks. Nonetheless, the gfxUnit must traverse all of these yMap entries on each scanline, wasting many memory cycles.

We handle this situation in the following manner: a special type of Cel is available that does nothing but load a new yMapBase pointer into the gfxUnit. This makes it possible to have more than one yMap for a given scene. Thus, when there are distinct regions on the screen, it is possible to specify a different yMap for each region.

To use yMap subroutines:

- 1. Set yMapBaseMaster and celsBaseMaster to default value in init routine.
- For subroutine, create yMap entry that points to a celBlock that has two cels in it. The first cel changes the yMapBase, and the second cel changes the celsBase.

Processing will proceed from the new yMapBase and the new celsBase. At the
end of the subroutine, use special celrecords to change the yMapBase and the
celsBase back to where they were (for the yMapBase, it is where you were
plus four bytes).

5.3.7.2 Converting Pixel Data to Legal Video Output Ranges

Example: JPEG images use pixel data with the following ranges:

Y = [0,255] Cr, Cb = [-128,127]

For the SOLO1 mode of operation, the software has to scale and shift every pixel to the legal video output ranges of:

Y = [16,235], where 16 is equal to zero Cr,Cb = [16,240], shifted such that 128 is equal to zero

This very time consuming operation is considerably reduced when the graphics engine is used. The pixel data can be put in the texture map in the original range. The luminance and chrominance scale is performed by using the globalAlpha parameter in the celRecord. For example an 85% scale is accomplished by setting the globalAlpha = 0xdb. Since no contribution from the background pixel is desired in this example, the alphaMode (2 bits of the celMode parameter of the celRecord) is set to "force background Alpha to zero." Alternatively, the alphaMode can be set to "one-minus-Alpha times the background color" where the background color (another celRecord parameter) is set to black.

Note: A luminance value of 0xff means "make this pixel transparent." Therefore, for the above example to work, all the luminance values of 0xff must be changed to 0xfe before storing in the texture map.

5.3.7.3 Solid Color Cels

A special mode has been included in the Graphics Engine to make the drawing of a solid color Cel very efficient.

- The texture will be one pixel of the desired color.
- As usual, the parameters: topLine, height, xLeftStart, dxLeft, xRightStart, and dxRight are used to define the boundaries of the cel.
- Set the celmode to DIR 4:2:2 or DIR 4:4:4.
- Set dux, duRowAdjust, dvx, dvRowAdjust = 0.

No matter how large the cel is, the gfxUnit will read the color once per line and write it many times into the scanline buffer.

Note: If any of the VQ modes are used for a solid color cel, the cel will draw, but it will be very inefficient from a memory bandwidth point of view. The gfxUnit will read the color from memory for every pixel even though it is the same color.

5.3.7.4 Reading a Texture from the WebTV Port

In order to use a texture from the WebTV Port, the texture must be copied to RAM. The gfxUnit can then source the texture from RAM.

SO	LO1	ASI	C
~		4+4+	•

Hardware Description

This chapter provides a functional description of the SOLO1 ASIC hardware. Each of SOLO1's primary logic groups is described, as well as the various clock domains, the reset logic, and the test modes.

SOLO1's primary logic groups are:

- CPU Bus Unit (busUnit)
- Main Memory Unit (memUnit)
- RIO Bus Interface Unit (rioUnit)
- Video Unit (vidUnit)
- Pixel Output Unit (potUnit)
- Graphics Engine (gfxUnit)
- Audio Unit (audUnit)
- I/O Device Unit (devUnit)
- Digital Video Input Unit (divUnit)
- Digital Video Encoder Unit (dveUnit)
- Video DAC Unit (dacUnit)
- Soft Modem Unit (modUnit)
- Smart Card UART (sucUnit)
- Miscellaneous/Clock Unit (mckUnit)
- Phase-locked Loop Unit (pllUnit)

These groups are illustrated in Figure 6-1.

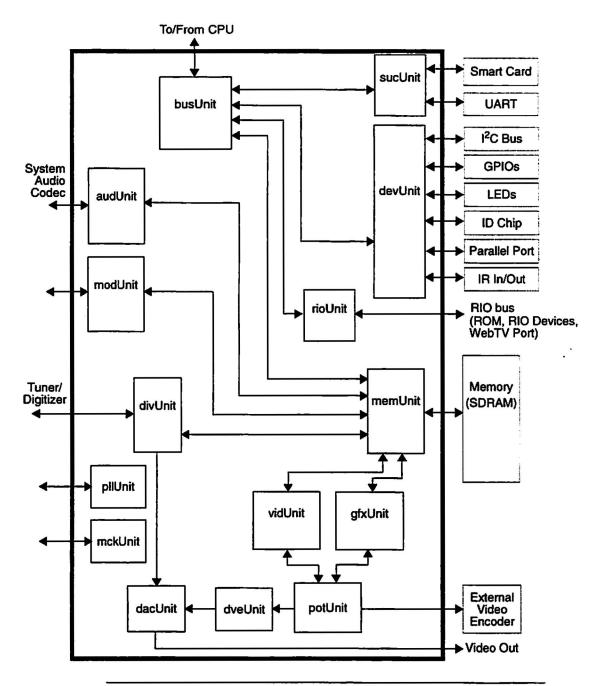


Figure 6-1 SOLOI Functional Block Diagram (only principal data paths shown)

6.1 Clock Domains

SOLO1 uses five clocks:

- SYS_CLK (83MHz)
- POT_CLK (2X/4X NTSC/PAL frequencies)
- AUD_CLK (11.89MHz)
- MOD_CLK (4.096MHz)
- DIV_LLC (2x NTSC/PAL frequency)

The SOLO1 ASIC is illustrated, in terms of clock domains, in Figure 6-2.

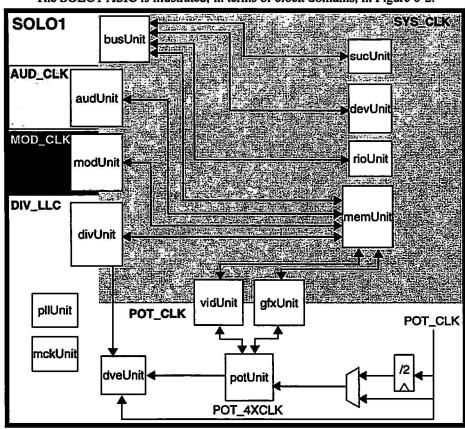


Figure 6-2 SOLO1 Clock Domains

Logic blocks that straddle two clock domains indicate different input and output timing parameters.

The POT_CLK and POT_4XCLK are generated in two different ways: If the internal video encoder is used, POT_CLK comes in at four times the pixel rate. This 4X clock goes directly to the on-chip video encoder and to the DACs. A 2X version is derived from this clock and sent to the potUnit.

When an external video encoder is used, POT_CLK comes in at 2X the pixel rate and is routed directly to the potUnit.

6.2 RIO Bus Unit

6.2.1 Overview

The RIO Bus Interface Unit (rioUnit) maintains the bidirectional data path between SOLO1 and up to two 32-bit wide ROM banks, the WebTV Expansion Port, and up to four ISA devices. A functional block diagram of the rioUnit logic is shown in Figure 6-3.

Typically, a WebTV system will contain one bank of Mask ROM and one bank of Flash ROM, although this configuration is not a requirement of the SOLO1 chip. The ROM interface supports either Mask ROM, Flash ROM, or SDRAM in either ROM bank, each of which can be up to 8MB. Standard and page-mode Mask/Flash ROMs are supported—burst-mode ROMs are not.

The rioUnit also communicates with up to four devices, using a protocol roughly corresponding to the ISA bus protocol. Each of these devices is mapped to a 2MB segment of memory and each has its own dedicated interrupt signal. Each device must be operated in slave mode, and no DMA to or from the device is permitted.

The rioUnit also maintains the WebTV Expansion Port interface, which in previous ASICs (SPOT and FIDO) was handled by the RAM interface (memUnit). The WebTV Expansion Port runs at a frequency divided down from the master system clock (WebTV Expansion Port clock = SYSCLK/4).

The rioUnit is completely de-coupled from the RAM interface, which permits each memory sub-system to be accessed simultaneously by different functional units within SOLO1. Typically, this means that the CPU can fetch code from the ROMs and service external ISA devices, while the graphics core manipulates video images in RAM.

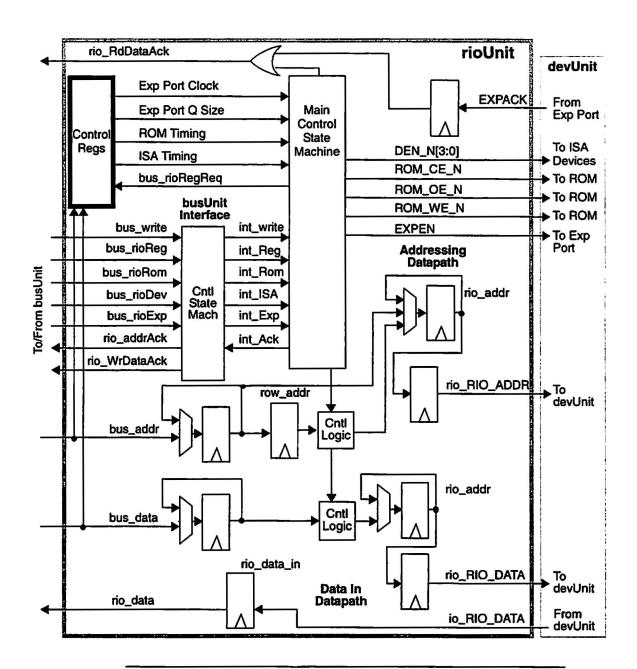


Figure 6-3 rioUnit Functional Block Diagram

6.2.2 Reset Behavior

During reset, the state of the RIO data bus is sensed by the rioUnit. The bus state is used to set the configuration parameters for both SOLO1 and the system. Among these parameters is the number of cycles it will take to perform a ROM access. The configuration bus indicates the system clock speed, the ROM access time, the ROM type (page mode), the ROM configuration, the memory access time, whether it's PAL or NTSC, and the general-purpose configuration parameters. Configurations are fully described in Chapter 5 "Software Description."

6.2.3 Arbitration

Only one of SOLO1's functional units is capable of accessing the rioUnit: the busUnit. The busUnit is the interface to the CPU, and as such, fetches code and data from both banks of ROM. Additionally, the CPU is responsible for servicing ISA devices, which are controlled from the rioUnit.

6.2.4 Control State Machine

The Control State Machine manages all of the rioUnit transactions. A diagram illustrating the state machine functions is shown in Figure 6-4. When the state machine moves from the *Idle* state to the *Start* state, it begins the transaction with the control, address and data information that is currently stored in the busUnit interface.

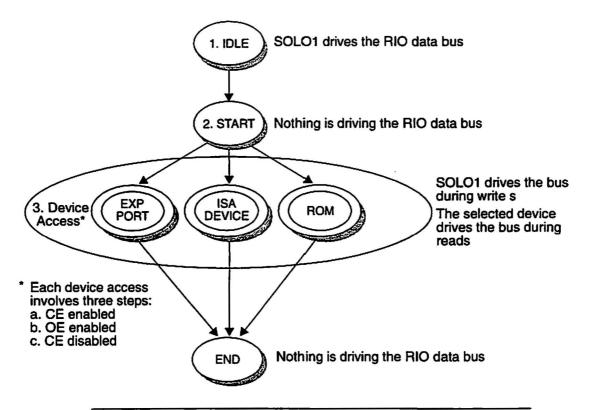


Figure 6-4 State Diagram of the rioUnit State Machine

6.2.4.1 State Machine Address Path

ROM and ISA-device addresses are passed straight through to the output pins connected to the devUnit. The 12-bit Expansion Port addresses are divided in half: the first clock drives the row address and the second clock drives the column address. The upper bits are used for control signals.

6.2.4.2 State Machine Data Path

When a transaction is initiated (i.e., when the state machine moves from the Idle to the Start state), the state machine accesses the data currently stored in the busUnit interface latches. ROM and Expansion Port data is passed through to the output pins. When writing to an ISA device, SOLO1 tristates the lower 16 bits and drives the data on the upper 16 bits.

6.2.5 RIO Bus Interface

The RIO bus provides the interface to the following system components:

- ROM
- modem
- ethernet controller
- hard disk drive
- · keyboard controller
- · serial port
- WebTV Expansion Port

The logical RIO bus interface consists of a 21-bit address bus and a 32-bit data bus. Output-enable and write-enable signals are shared among all memory parts, while there are two separate chip-enable signals that allow access to two separate 8MB banks of ROM. Data is accessed from the memory on 4-byte boundaries only.

There are two banks of output enables: CE0-1 and DE0-3. CE0-1 control the system ROM and DE0-3 enable up to four ISA devices. Bank selection is determined by OR'ing the six enable signals with ALT_DEV_EN.

Timing for a write to ROM is shown in Figure 6-5. The timing for a read transaction is identical, except that the states of the WE_N and OE_N signals are swapped. The times represented by t0 (cycle time) and t1 (strobe off) are determined by the ROM specifications. 'Turnaround Time' is a programmable value that determines how much time must elapse between the end of one bus transaction and the start of another.

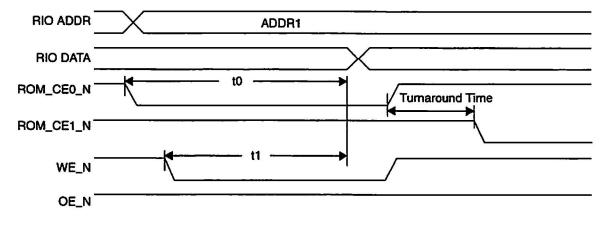


Figure 6-5 RIO Bus Timing for a ROM Write

6.2.6 ISA Device Interface

The logical ISA bus interface consists of a 19-bit address bus and a 32-bit data bus, though in practice, only the lowest 16 bits of the data bus are used. Some of the sixteen upper bits are used for control signals.

ISA devices share the same physical address and data buses used by the system ROM interface. Up to four ISA devices are currently supported, each with it's own device enable signal (DEO_N - DE3_N) and it's own read and write enables (IOR_N and IOW_N, respectively).

An additional four ISA devices can be supported. These devices use the same 8MB of address space as the first four devices. They are addressed by OR'ing a signal similar to ALT_DEV_EN, in the busUnit interface, with the existing enable lines.

There are also two additional memory enables (MEMR_N and MEMW_N) that are used to interface to any optional, external memory accompanying the ISA devices. Enables are directed to either the ISA device itself, or to its external memory, by OR'ing the device enables with the memory enables.

Timing for a typical write to an ISA device is shown in Figure 6-6. The timing for a read transaction is identical, except that the states of the IOW_N and IOR_N signals are swapped.

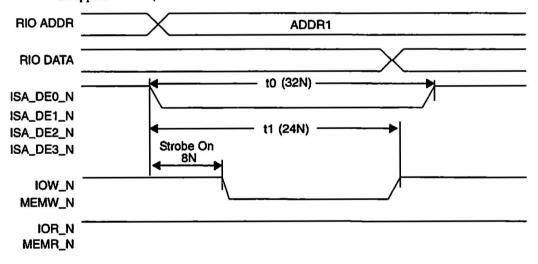


Figure 6-6 RIO Bus Timing for ISA Reads and Writes

6.2.6.1 Holding the Bus for a Slow Device

If the rioUnit initiates a read or write transaction with an ISA device, and the data is not ready, the enable signals can be 'stretched' until the data arrives. The signal that controls the transaction timing is IOCHRDY. Pulling IOCHRDY low effectively freezes the enable signals and ties the entire bus up until the data arrives. There are no escapes once IOCHRDY is pulled low. The timing for this operation is shown in Figure 6-7.

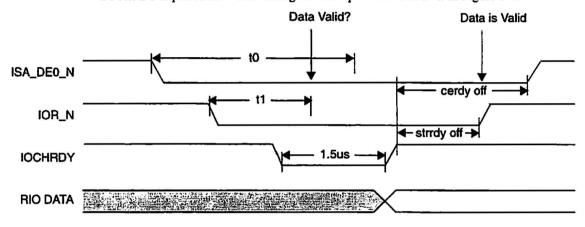


Figure 6-7 Hold the Bus!

6.2.7 WebTV Port Expansion Interface

The logical WebTV Expansion Port interface consists of an 11-bit multiplexed address bus and a 32-bit data bus. Address and data buses are shared with the corresponding ROM and ISA interfaces. Additionally, the Expansion Port interface contains row and column address strobes and a few other SDRAM control signals. These control signals are shared with high-order address bits on the ROM bus, that are unused with the WebTV Expansion Port protocol.

6.2.8 Signal Definitions

The rioUnit signals are described in Table 6-1.

Table 6-1 rioUnit Signals

Signal Name	Direction	Description
RIO_CEO_N	0	ISA device select
RIO_CE1_N	0	ROM bank select
RIO_OE_N	0	ROM output enable
RIO_WE_N	0	ROM write enable
RIO_ADDR[22:2]	0	ROM address

Table 6-1 rioUnit Signals (Continued)

Signal Name	Direction	Description
RIO_DATA[31:0]	I/O	ROM data
RIO_DEN_N[3:0]	0	RIO expansion device enables
RIO_DINT[3:0]	I	RIO expansion interrupts
RIO_DEVIORDY	Ī	RIO expansion I/O ready
RIO_EXPCLK	0	RIO expansion device clock
RIO_EXPEN	0	Memory expansion bus enable
RIO_EXPACK_N	I	Memory expansion acknowledge
IOR_N	0	ISA Device Read Signal
IOW_N	0	ISA Device Write Signal
MEMR_N	0	ISA Device External Memory Read Enable
MEMW_N	0	ISA Device External Memory Write Enable
ALT_DEV_EN	0	Alternate Device Enable- selects between ISA and ROM enables
ROM_RESET_N	I	rioUnit Async Reset Signal
ISA_DE_N [3:0]	0	ISA Device Enable Signals

6.3 Audio Unit

6.3.1 Overview

The Audio Interface Unit (audUnit) maintains serial data communication between SOLO1 and an off-chip audio DAC or combined DAC/ADC (codec).

Note: While SOLO1 supports both discrete DACs and combined DAC/ADCs, the fact that the DAC and the ADC must share the same clock and synchronization signals, combined with the cost, lean towards a combined DAC/ADC configuration.

Output audio samples are read from RAM through a DMA channel, and sent as a single-bit, serial data stream to the audio DAC in 16-bit stereo samples. SOLO1's audUnit supports 8-bit/16-bit, stereo/mono samples for audio output. Samples are converted by the audUnit into 16-bit stereo format for the DAC. DMA chaining allows efficient manipulation of audio data with minimal interaction by the CPU.

Audio input samples from the ADC are 16-bit stereo. Input samples are not converted by the audUnit into any other format (i.e. 8-bit and mono inputs are not supported). Input samples are written to RAM through a DMA channel.

The audUnit provides local buffering for both DMA channels (read and write) to minimize memory bus bandwidth used for audio transactions.

The audUnit also supports two different sample clock frequencies in a single system. Supported frequencies are 44.1kHz and 48kHz—the two predominant high-quality digital audio sampling frequencies. Both the input and output audio paths must use the same sampling frequency.

A functional block diagram illustrating the audUnit logic is shown in Figure 6-8.

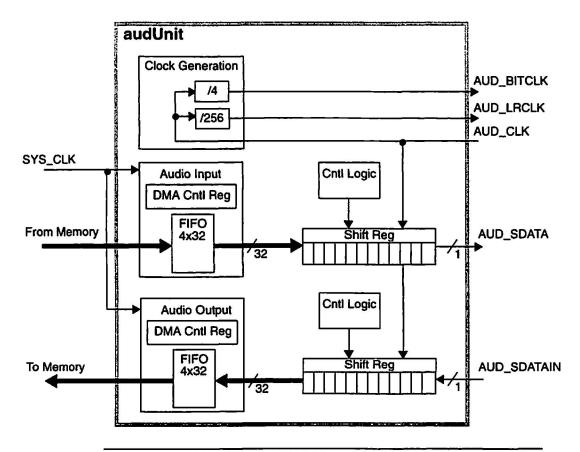


Figure 6-8 audUnit Functional Block Diagram

6.3.2 Internal vs. Codec Data Formats

The internal and codec formats for audio data are compared in Figure 6-9.

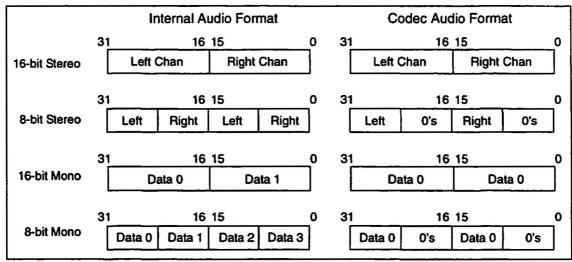


Figure 6-9 Internal and Codec Audio Data Formats

6.3.3 Data Out Formats

The audUnit supports the following DAC/ADCs:

- AKM 4520 (Data Out = Right-aligned, Data In = Left-aligned)
- Burr-Brown PM3001 (Data Out = Right-aligned, Data In = Left-aligned)
- AKM 4532 (Data Out = Left-aligned, Data In = Left-aligned)

The Right- and Left-aligned data formats are illustrated in Figure 6-10.

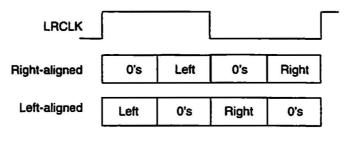
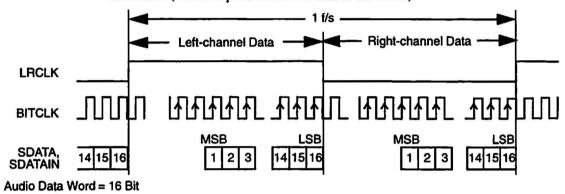


Figure 6-10 Right/Left-Aligned Data Formats

6.3.4 Data Interface Formats

SOLO1's digital audio input and output interfaces are implemented using four signals:

- LRCLK (left/right channel clock)
- BITCLK (bit clock)
- SDATA (audio input from the codec to the audUnit)
- SDATAIN (audio output from the audUnit to the codec)



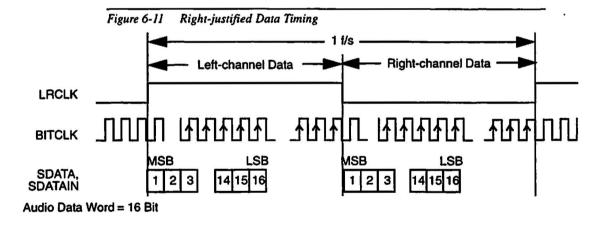


Figure 6-12 Left-justified Data Timing

The Bit Clock (BITCLK) and the Left/Right Clock (LRCLK) are derived from the audio master clock (AUD_CLK) and are driven by SOLO1. The BITCLK frequency is AUD_CLK/4 and the LRCLK frequency is AUD_CLK/256. Both digital input and output paths must use the same clock signals.

6.3.5 DMA Engine Description

The audUnit DMA engines support memory buffers of up to 64KB. Each buffer must be aligned on a 32-byte boundary.

DMA read buffers (since they may contain 8-bit mono samples) may be any arbitrary integer size. DMA write buffers, which only contain 16-bit stereo samples, must be a multiple of 32 bytes in size.

Each DMA engine uses buffer chaining, which requires minimum CPU interaction. Audio sample data is read from the current output buffer. When the current buffer is empty, and if there is a valid "next" buffer in the chain, then the "next" buffer becomes the "current" buffer and DMA transactions continue - allowing the CPU to refill the now empty buffer and re-insert it in the chain as the new "next" buffer.

The write DMA engine performs similarly in the opposite direction—i.e. when the current buffer is full, DMA transactions continue to the valid "next" buffer.

6.3.6 Signal Definitions

The audUnit's external and internal signals are described in the following table.

Table 6-2 audUnit Signals

Signal Name	Direction	Description
io_SYS_2XCLKIN	I	Main system clock
io_AUD_CLK	I	Audio clock
io_AUD_SDATAIN	I	Audio shift data input
io_MOD_CLK	I	Modem clock
io_MOD_SDATAIN	I	Modem shift data input
aud_AUD_BITCLK	0	Audio bit clock
aud_AUD_LRCLK	0	Audio left/right clock
aud_AUD_SDATA	0	Audio shift data output
mod_MOD_BITCLK	0	Modem bit clock
mod_MOD_LRCLK	0	Modem left/right clock
mod_MOD_SDATA	0	Modem shift data output
aud_diagBus[5:0]	0	Audio debug bus
mod_diagBus[5:0]	0	Modem debug bus
mck_audUnit_areset_n	I	Async SYS_2XCLKIN reset signal
mck_modUnit_areset_n	I	Async SYS_2XCLKIN reset signal
ckd_audUnit_aresetAud_n	I	Async AUD_CLK reset signal

Table 6-2 audUnit Signals (Continued)

Signal Name	Direction	Description
ckd_modUnit_aresetMod_n	I	Async MOD_CLK reset signal
test_syncInitClocks	Ī	Initialize free running clocks
bus_write	I	Transaction from busUnit is write
bus_addr[6:2]	I	busUnit address bus
bus_data[25:0]	I	busUnit data bus
mem_data[31:0]	1	memUnit data bus
bus_audRegReq	I	Transaction request from busUnit
mem_audRAddrAck	I	Address ack from memUnit
mem_audWAddrAck	I	Address ack from memUnit
mem_audRdAck	I	Read data ack from memUnit
mem_audWrAck	I	Write data ack from memUnit
bus_modRegReq	I	Transaction request from busUnit
mem_modRAddrAck	I	Address ack from memUnit
mem_modWAddrAck	I	Address ack from memUnit
mem_modRdAck	I	Read data ack from memUnit
mem_modWrAck	I	Write data ack from memUnit
aud_dmaInInt	Ö	End of Input DMA interrupt
aud_dmaOutInt	0	End of Output DMA interrupt
aud_busRegAck	0	General ack to busUnit
aud_memRReq	0	Read transaction request to memUnit
aud_memWReq	0	Write transaction request to memUnit
aud_memDqm[3:0]	0	Byte write mask controls
aud_rAddr[25:2]	0	audUnit read address bus
aud_wAddr[25:2]	0	audUnit write address bus
aud_busRegData[25:0]	0	busUnit register data
aud_data[31:0]	0	Memory write data
mod_dmaInInt	0	End-of-Input DMA interrupt
mod_dmaOutInt	0	End-of-Output DMA interrupt
mod_busRegAck	0	General acknowledge to busUnit
mod_memRReq	0	Transaction request to memUnit
mod_memWReq	0	Transaction request to memUnit

Table 6-2 audUnit Signals (Continued)

Signal Name	Direction	Description
mod_memDqm[3:0]	0	Byte write mask controls
mod_rAddr[25:2]	0	audUnit read address bus
mod_wAddr[25:2]	0	audUnit write address bus
mod_busRegData[25:0]	0	busUnit register data
mod_data[31:0]	0	memUnit write data

6.4 Bus Unit

6.4.1 Overview

The Bus Unit (busUnit) is the interface between the CPU and SOLO1, and provides efficient access to both code (typically stored in ROM) and data (typically stored in RAM), as well as various memory-mapped devices.

A functional block diagram of the busUnit logic is shown in Figure 6-12.

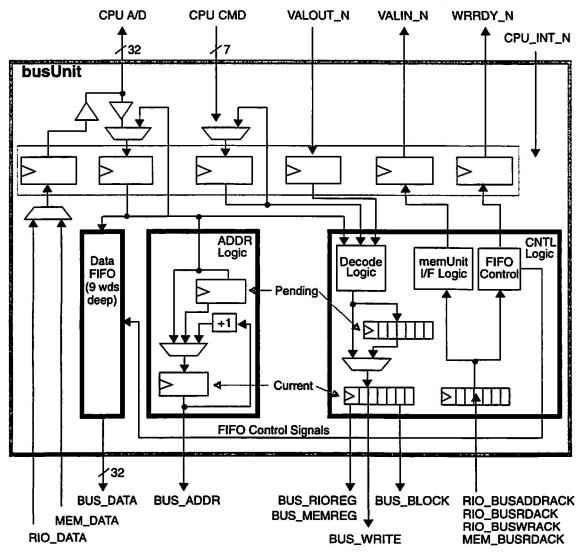


Figure 6-13 busUnit Functional Block Diagram

The SOLO1 CPU interface is a 32-bit multiplexed address and data bus supporting the R4640 and R5230 implementations of the MIPS architecture. Single-word, as well as both 4- and 8-word burst transactions are supported. SOLO1 is also able to arbitrate the CPU bus for systems containing other master/slave devices.

SOLO1 uses the R4XXX-compatible write protocol. The write data pattern is 1word/1 MasterClock cycle. 32-byte read and write bursts are supported for cache fill and write-back transactions. Burst read transactions are returned in sub-block order. Burst write transactions are handled in linear order.

Note that the data checking feature of the R4640 (parity) is not used by SOLO1.

6.4.2 Reset Behavior

All interrupts are disabled at reset and all interrupt status bits are cleared, although interrupt events will be caught regardless of the values of their enable bits. Such events can be observed by polling the status registers that are not masked by enable bits. Or, if the enable bit is subsequently set, a prior interrupt event will immediately cause a CPU interrupt.

- · All fences are in a disabled state at reset.
- The watchdog timer is disabled at reset. Its software reset (clear) value is set to 0x00 at reset.
- ROM address space aliases onto RAM address space at reset. This must be disabled with the CHPCNTL register.

6.4.3 Write Buffer

The busUnit interface contains a 32-byte write buffer used exclusively to maintain efficient use of the memory buses during write operations.

Burst write transactions are collected by the write buffer and held until the memory interface unit is able to accept the write. Burst write transactions are only supported to the RAM.

The write buffer may be used to collect consecutive single-word write transactions while waiting for the memory interface to be able to accept them (i.e., non-cached writes may be stacked up in the write buffer while write transactions to Flash memory or other slow devices are underway).

6.4.4 Signal Definitions

The busUnit's interface signals are defined in Table 6-3.

Table 6-3 busUnit Interface Signals

Signal Name	Direction	Description
CPU_AD	ΙΟ	CPU address and data bus
CPU_CMD	I/O	CPU command bus
CPU_VALOUT_N	I	Address-, command-, and data-valid strobe
CPU_VALIN_N	0	Data and command valid back to CPU
CPU_WRRDY_N	0	CPU write-ready signal
CPU_INT_N	0	CPU interrupt line
bus_memReq	0	busUnit Memory Request
bus_addr [25:2]	0	busUnit Transaction Address
bus_size	0	busUnit Transaction Size (4 or 16B)
bus_write	0	busUnit Transaction Write Signal
bus_data [31:0].	0 ,	busUnit Data Output
mem_busAck	I	memUnit Ack to busUnit
mem_data [31:0]	I	memUnit Data Output
mem_expAck	0	Expansion Port Acknowledge
RIO_ADDR [22:2]	0	ISA/ROM/Expansion Port address bus
RIO_DATA [31:0]	ΙΛΟ	data to/from RIO bus
RIO_CE [1:0]	0	ROM 0/1 chip enable
RIO_DEN_N [3:0]	0	ISA device (0-3) enables
RIO_DEVIORDY	I	ISA IOCHRDY
RIO_DINT [3:0]	I	ISA device (0-3) interrupts
RIO_EXPACK_N	1	Expansion Port acknowledge
RIO_EXPCLK	0	Expansion Port clock
RIO_EXPEN	0	Expansion Port enable
bus_busError	I	Bus Error Signal/Cancel Transaction
mck_reset	I	mckUnit Reset Signal
bus_rioRegReq	1	Request transaction with RIO control registers
bus_memRegReq	I	Request transaction with memUnit control registers

Table 6-3 busUnit Interface Signals (Continued)

Signal Name	Direction	Description
RIO_BUSADDRACK	I	RIO bus address acknowledge
RIO_BUSRDACK	Ī	RIO bus read acknowledge
RIO_BUSWRACK	Ī	RIO bus write acknowledge
MEM_BUSRDACK	I	Memory bus read acknowledge

6.4.5 Bus Reads

A bus read operation is comprised of the following events:

- An address is placed on the CPU Address and Data bus.
 The CPU issues read and size commands, and asserts VALOUT_N.
- 2. The address is latched and decoded by the busUnit.
- 3. The address and decode logic will do one of the following:
 - a) hold and wait
 - b) save the pending address
 - c) start the read operation specified by the current address
- 4. If 'c,' then the request and control signals are asserted. When the bus address is valid, the read is performed. The busUnit sends a read-acknowledge signal to the CPU. Go to Step 7.
- 5. If 'b,' then the busUnit waits for the current operation to complete. Return to Step 4.
- 6. If 'c,' then the busUnit waits for the pending address to become current. Return to Step 5.
- The busUnit asserts VALIN_N.

6.4.6 Bus Writes

A bus write operation is comprised of the following events:

- An address is placed on the CPU Address and Data bus.
 The CPU issues write and size commands, and asserts VALOUT_N.
- The address is latched and decoded by the busUnit.
- 3. Data is loaded into the Data FIFO.
- 4. The address and decode logic will do one of the following:
 - a) hold and wait
 - b) save the pending address
 - c) start the write operation specified by the current address

- If 'c,' then the request and control signals are asserted. When the bus address is valid, the write is performed. The busUnit sends a write-acknowledge signal to the CPU. Go to Step 8.
- 6. If 'b,' then the busUnit waits for the current operation to complete. Return to Step 5.
- 7. If 'c,' then the busUnit waits for the pending address to become current, return to Step 6.
- 8. The busUnit asserts VALIN_N.

6.5 Digital Video Input Unit

6.5.1 Overview

The Digital Video Input Unit (divUnit) captures both audio and video digital data from the external audio/video decoders. The video data can be cropped, re-sampled at a variety of ratios, and decimated before it's sent to the memUnit. The frames deposited in memory may be sub-sampled to 1/4 screen to reduce bandwidth for PIP-type applications. Alternatively, the video data can be passed straight through the resampling logic and DMA'd directly to the memUnit.

The audio data path converts the incoming serial data stream to a 16-bit bus before the DMA to the memUnit.

Figure 6-14 illustrates the significant components of the divUnit. divUnit --- Video Data Path **DIV_DATA** [7:0]

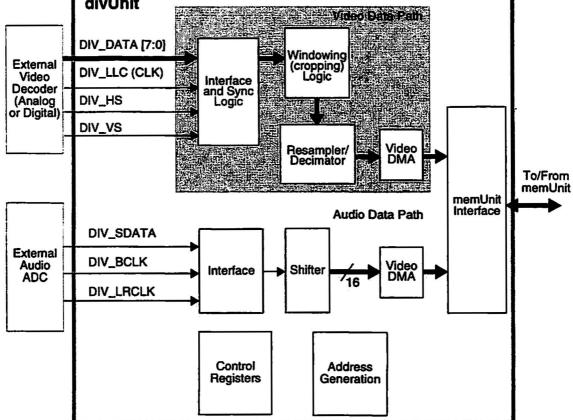


Figure 6-14 divUnit Functional Block Diagram

The divUnit is designed to connect to video devices supporting a 16-bit data path. The synchronization signals may be supplied as separate pins (DIV_HS, DIV_VS) or as 4-byte codes embedded in the video stream, as in SMPTE 259M. If separate pins are used, either the rising or falling edge of the signals can be selected as reference points.

The divUnit can shift data, flip the MSB and LSB, take the 2's complement of a value, flip the right and left channels, and change the beginning of the data sample relative to the clock.

The windowing logic crops the incoming video image via a set of control registers that define the active video region with respect to the selected timing references. LBORDER defines the number of 2x pixel clocks from the horizontal reference to the first pixel on a line. TBORDER defines the number of lines from vertical sync to the first active video line. HEIGHT and WIDTH define the picture size.

To support non-square pixel devices that may be cheaper than square alternatives, a resampler is included that supports the following sampling ratios:

- 720 -> 640
- 720 -> 768
- 720 -> 320
- 720 -> 384
- 1 -> 1
- 2 -> 1

Following the resampling stage, a decimator is provided that supports $1 \rightarrow 1$ and $2 \rightarrow 1$ ratios. The $1 \rightarrow 1$ ratios available in both the resampler and decimator are used to pass the data through without any modification.

Note that the registers that define the active region describe the horizontal area after resampling, should it be enabled.

The video data is then deposited into main memory via DMA. The base address and row-to-row stride are specified in the BASEADDR and STRIDE registers. The stride is specified in memory bytes. To reduce bandwidth requirements when the desired effect is to reduce the area covered by video to around 1/4 screen or less, the images may be decimated to half horizontal resolution prior to DMA. A suitable 1/4 screen picture can be created by selecting only one field per frame for capture.

6.5.2 Signal Definitions

The following table lists the divUnit interface signals.

Table 6-4 divUnit Interface Signals

Signal Name	Input/ Output	Signal Description
DIV_DATA[7:0]	I	Data port
DIV_BCLK	I	Audio bit clock
DIV_VS	I	Vertical sync (control signal)
DIV_HS	I	Horizontal sync (control signal)
DIV_LLC	Ĭ	Video clock
DIV_LRCLK	I	Audio left-right channel clock
DIV_SDATA	I	Audio serial stream
div_memAddr[31:0]	0	Memory address for video DMA
div_memReq	0	Memory Request
mem_divAddrAck	I	Memory controller address ack
div_memWriteData[31:0]	Output	Data path to memory
mem_divDataAck	Input	Write data acknowledge

6.5.3 Programming Model

This section lists the divUnit's control registers. Detailed descriptions of these registers are provided in Chapter 5 "Software Description."

Table 6-5 divUnit Control Registers

Register Name	Description	
DIV_SYNCSEL	Select sync source (0 - pins, 1 - embedded	
DIV_INVHS	Invert sense of HS source	
DIV_INVVS	Invert sense of VS source	
DIV_VSISFLD	Use VS as field indicator	
DIV_FIXSYNCERRS	Enable forward error correction on embedded syncs	
DIV_DMAEN	Enable DIVIT DMA	
DIV_NV	Next Valid	
DIV_NVF	Next Valid Forever	
DIV_NFIELD	Next field select	
DIV_NTBORDER[9:0]	Next Top Border, inclusive	

Table 6-5 divUnit Control Registers (Continued)

Register Name	Description	
DIV_NLBORDER[9:0]	Next Left Border, in 2x pixel clocks, inclusive	
DIV_NRBORDER[9:0]	Next Right Border, in 2x pixel clocks, inclusive	
DIV_NBBORDER[9:0]	Next Bottom Border, inclusive	
DIV_NRSMPMODE[1:0]	Next Resampler Mode	
DIV_NDECEN	Next Decimator Enable	
DIV_NSTART[22:5]	Next Starting Address (32B aligned)	
DIV_NSTRIDE[15:5]	Next line-to-line stride for next buffer (32B)	
DIV_CFIELD	Current field select	
DIV_CTBORDER[9:0]	Current Top Border, inclusive	
DIV_CLBORDER[9:0]	Current Left Border, in 2x pixel clocks, inclusive	
DIV_CRBORDER[9:0]	Current Right Border, in 2x pixel clocks, inclusive	
DIV_CBBORDER[9:0]	Current Bottom Border, inclusive	
DIV_CRSMPMODE[1:0]	Current Resampler Mode	
DIV_CDECEN	Current Decimator Enable	
DIV_CSTART[22:5]	Current Starting Address (32B aligned)	
DIV_CSTRIDE[15:5]	Current line-to-line stride for next buffer (32B)	

6.6 Modem Unit

6.6.1 Overview

The Modern Interface Unit (modUnit) maintains serial data communication between SOLO1 and an off-chip modern.

Output data is read from RAM through a DMA channel, and sent as a single-bit, serial data stream to the modem in 16-bit mono samples. Note that the modUnit only supports 16-bit mono data. DMA chaining allows efficient manipulation of the data with minimal interaction by the CPU. Input data is written to RAM through a DMA channel.

The modUnit provides local buffering for both DMA channels (read and write) to minimize memory bus bandwidth used for transactions.

A functional block diagram illustrating the modUnit logic is shown in Figure 6-15.

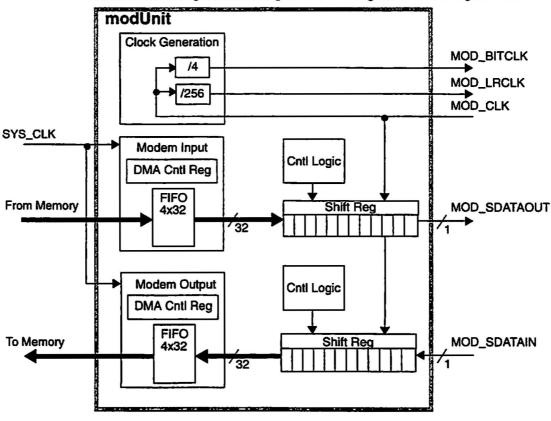


Figure 6-15 modUnit Functional Block Diagram

6.6.2 Signal Definitions

The modUnit is integrated with the audUnit, and has no separate signals. refer to the audUnit section, earlier in this chapter, for the signal descriptions that apply to the modUnit.

6.7 I/O Device Unit

6.7.1 Overview

The I/O Device Unit (devUnit) is comprised of four principal subunits:

- devRegs—provides control registers for the busUnit interface, the LEDs, the chip ID, the GPIO lines, the I²C bus, the graphics RAM delay, and the internal debug bus
- · devPPort-controls the Parallel Port
- · devIRIn-controls the IR input
- · devIROut-controls the IR output

A functional block diagram supporting the devUnit logic is shown in Figure 6-16.

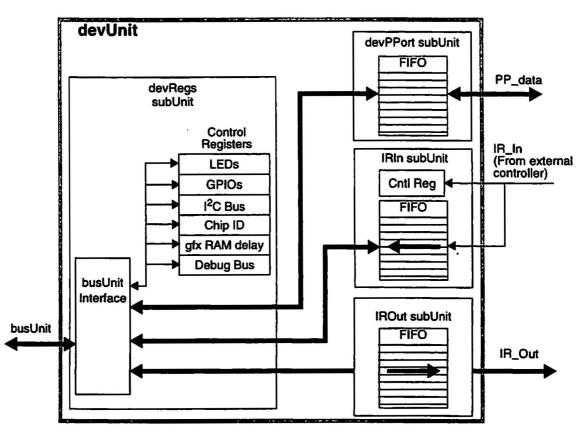


Figure 6-16 devUnit Functional Block Diagram

6.7.2 Signal Definitions

The following table lists the devUnit interface signals.

Table 6-6 modUnit Interface Signals

Signal Name	Input/ Output	Signal Description
io_SYS_2XCLKIN	I	System clock
mck_resct_n	I	System reset
io_IR_IN	I	Infrared input data
dev_IR_CLK	0	Clock for external IR controller
dev_IR_CLK_en	0	Enable for external IR controller
io_IR_CLK	I	Infrared input clock
dev_IR_OUT	0	Infrared output data
dev_IR_OUT_en	0	Infrared output data enable
io_IR_OUT	Ī	Infrared output read data
dev_MISC_LED[2:0]	0	LED value
io_ID_DATA	I	ID data input
dev_ID_DATA	0	ID data output
dev_ID_DATA_en	0	ID data output enable
io_IIC_CLK	I	EEPROM clock in (reg)
dev_IIC_CLK	0	EEPROM clock (reg)
dev_IIC_CLK_en	0	EEPROM clock enable (reg)
io_IIC_DATA	I	EEPROM data in (to reg)
dev_IIC_DATA	0	EEPROM data (reg)
dev_IIC_DATA_en	0	EEPROM enable (reg)
io_GPIO[15:0]	I	GPIO inputs
io_GPIO_r2[15:0]	0	Synchronized GPIO inputs
dev_GPIO[15:0]	0	GPIO outputs
dev_GPIO_en[15:0]	0	GPIO output enables
io_parametric	I	Parametric chain output
test_paramTestMode	I	Parametric test mode
test_scanMode	I	Scan enable
dev_ramDelay[7:0]	0	Internal RAM input timing
io_PP_DATA[7:0]	I	Parallel port input data

Table 6-6 modUnit Interface Signals (Continued)

Signal Name	Input/ Output	Signal Description
io_PP_FAULT_N	I	Parallel port fault
io_PP_SELECT	I	Parallel port select
io_PP_ACK_N	I	Parallel port acknowledge
io_PP_BUSY	I	Parallel port device busy
io_PP_ERROR	I	Parallel port device error
dev_PP_AUTOFD_N	0	Parallel port auto feed output
dev_PP_INIT_N	0	Parallel port initialize
dev_PP_STROBE_N	0	Parallel port strobe
dev_PP_SELIN_N	0	Parallel port select
dev_PP_DATA[7:0]	0	Parallel port output data
dev_PP_DATA_en	0	Parallel port output enable
dev_PP_DIR	0	Parallel port direction
bus_devReq	I	busUnit request for devUnit
bus_write	I	busUnit write signal
bus_addr[11:2]	I	busUnit address bus
bus_data[31:0]	I	busUnit data bus
div_pwmEnable	I	DIVIT pulse-width mod enable
div_pwm	I	DIVIT pulse-width mod
dev_busAck	0	Acknowledge to busUnit
dev_irInt	0	IR receiver interrupt
dev_irOutInt	0	IR blaster interrupt
dev_data[31:0]	0	devUnit data bus
dev_pportInt	0	Parallel Port interrupt
dev_diagSel[3:0]	0	Select diagnostics information
dev_diagEnable	0	Enable the debug bus
dev_diagBus[5:0]	0	Debug bus

6.7.3 Parallel Port

SOLO1 uses the same Parallel Port control logic that is used by the Sasha ASIC. This section describes the changes and modifications that differentiate SOLO1's integral parallel port interface from Sasha.

Table 6-7 Port Modes

ASIC	Software	Compatibility Fifo	ECP-F	ECP-R
Sasha	Yes	Yes (with bug)	Yes	No
SOLO1	Yes	Yes	Yes	Yes

6.7.3.1 Differences Between the SOLO1 and Sasha Handshake State Machines

This state machine controls all modes except software mode:

- TOutCountEn was removed from SOLO1. The TimeOut register that was associated with this feature has changed from a time-out register value to a bit that creates an immediate cycle abort. See #6 in "Register Changes."
- To support the reverse channel in ECP mode, SOLO1's state-machine gained the
 ability to control the HWStb_ signal. In addition, to enable the use of FIFO's in this
 direction, a ReversePush signal was created to load the FIFO from the peripheral.
 The associated states and the way in and out were implemented.
- The ECP_TERM state in Sasha pointed to no other state. If software were to enable
 the hardware ECP mode, the only way to get control and switch modes would be
 through a parallel port reset. Now in SOLO1, ECP_TERM returns the port to the idle
 state.
- 4. The Compatibility FIFO mode in Sasha adhered strictly to the 1284 spec for the handshake. To use real-life printers (like the HP series), SOLO1 was modified to handshake in a manner compatible with both the IEEE1284 and Centronics specs.

6.7.3.2 Register Changes

- DATA—Can read from FIFO when in ECP-R mode.
- 2. SWInit_ in the CTRL register selects -F or -R in ECP mode.
- STAT register bits 2 and 4 were switched in SOLO1. This is the Fault_ and Error mixup. On the Sampson boards, the DB-25 was mis-wired and software re-labeled the
 bits in their spec to match the physical hardware. By correcting the LC2 layout and
 swapping the bits in SOLO1, software sees no change.
- 4. STAT2 was added to allow software to read the output that SOLO1 is generating. This is useful to help determine what state the hardware is in if a peripheral fails in the middle of a cycle. Software can decide to abort if the machine is in a transitory state

- too long and the external device has broken the specification (i.e., if the cable is unplugged).
- 5. FifoOvf in FIFOSTAT could only be read as 0' and was already implemented correctly in another register (FifoOvfInt in IST). It was removed.
- 6. TIMEOUT was the time-out count until an auto abort by the hardware, should something go wrong. Now software can decide if something went wrong and create it's own abort. Bit 0 of the TIMEOUT register will not cause an ECP abort whenever a '1' is written to it.
- 7. IEN bits 1 and 5 were removed from SOLO1. Bit 1 was FifoOvfIntEn. We will never write data to a full FIFO. That would be a software bug. Bit 5 is gone because of the TIMEOUT change in #6.
- 8. IST, bits 1 and 5 gone again for same reasons as above.
- 9. CLRINT, bits 1 and 5 gone again for same reasons as above.
- 10. Sasha had a general register elsewhere that could enable and disable the port. SOLO1 has a new ENABLE register that does this function.

6.7.3.3 Other Changes

- 1. The 750ns timer has been scaled to match SOLO1's clock.
- Several output signals in Sasha were gated through a hand-instantiated glitchless MUX. These were removed from SOLO1 and all outputs are registered to avoid glitches (and meet timing).
- Sasha had RAMs or something instantiated for the FIFOs while SOLO1 has the latchbased FIFOs.
- The SOLO1 devPPort does not use the bus interface state-machine from Sasha. It
 instead interfaces to the devUnit.
- 5. If external buffer drivers are used on Sasha, there is a chance for data contention. SOLO1 generates a four-cycle margin of safety between direction changes.
- Synchronizers on Sasha inputs were added late and thus hand-instantiated.Synchronizers were added early on SOLO1 and are not hand-instantiated.

6.8 Pixel Output Unit

6.8.1 Overview

The Pixel Output Unit (potUnit) receives data from both the vidUnit and gfxUnit, and outputs it to either the SAA7187 external DAC, or to the dveUnit. Figure 6-17 provides a functional block diagram of the potUnit logic.

The gfxUnit and vidUnit provide both 16- and 24-bit data words. The potUnit selects between the two 16-bit data sources, aligns the chosen 16-bit word, and sends it to the SAA7187 at the rate of one byte per clock cycle. The data alignment is necessary to adjust the internal pipeline to the external DAC.

The potUnit also selects between the two 24-bit data sources and passes the chosen 24-bit value through the potUnit at the rate of 24 bits every other clock cycle.

Vid_WidePixOut is the averaged color value from the vidUnit. gfxWidePix is the

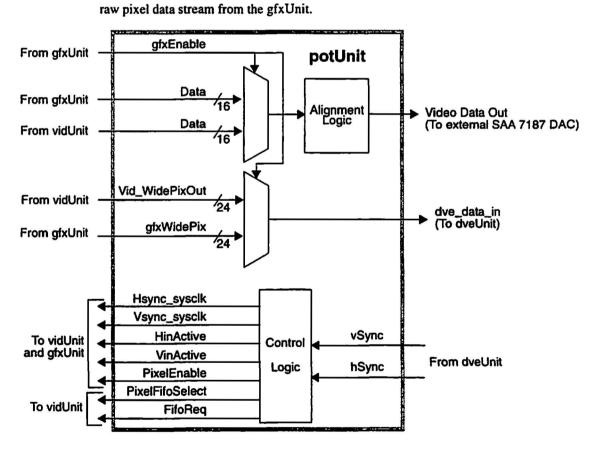


Figure 6-17 potUnit Functional Block Diagram

The potUnit also contains logic that aligns the horizontal and vertical syncs to the system clock, and passes them on to both the gfxUnit and vidUnit. This logic also controls the vidUnit's FIFOs: A select line (PixelFifoSelect) ping-pongs between the two video FIFOs to retrieve the next available data from the vidUnit; the FIFO reload line (FifoReq) instructs the vidUnit to reload a FIFO when all of its eight entries have been read.

6.8.2 Signal Definitions

The following table lists the potUnit interface signals.

Table 6-8 potUnit Interface Signals

Signal Name	Input/ Output	Signal Description
VID_VSYNC_N	NO	Vertical sync input
VID_HSYNC_N	ΙΛΟ	Horizontal sync input
POT_CLK	ΝO	Video display pixel clock
VID_DATA[7:0]	ΙΛΟ	Video data and configuration bus
VidDataWide	ΙΟ	Averaged pixel data
Vsync_sysclk	0	Vertical sync aligned to the system clock
Hsync_syscik	0	Horizontal sync aligned to the system clock
HinActive	0	Current horizontal position is within the screen's active region
VinActive	0	Current vertical position is within the screen's active region
PixelEnable	0	Requests more data from vidUnit or gfxUnit
PixelFifoSelect	0	Selects vidUnit's video FIFO entry
VID_DATA_OUT	0	Data from vidUnit to external DAC
FifoReq	0	Reloads vidUnit's video FIFOs
use_gfx	I	Selects data from gfxUnit

6.9 Video Unit

6.9.1 Overview

The Video unit (vidUnit) receives 32-bit video data from memory and outputs it as either 16- or 24-bit words. Sixteen-bit data is sent to the potUnit and 24-bit data is sent to the dveUnit.

Incoming data is stored in a 12x32 DMA FIFO that is separated into three 4-word sections. When the DMA FIFO has four words or less remaining, eight more words are loaded from memory.

One half of each 32-bit word is read from the DMA FIFO and input to a pair of 8x16 video FIFOs. The output of these FIFOs will either leave the vidUnit as a pair of 16-bit words, or as a single, 24-bit word. The 16-bit words are sent to a MUX, where they are transferred to the potUnit. The 24-bit data word is the product of "averaging" the pixels represented by four, 16-bit words, and is sent to the dveUnit via a second MUX.

The vidUnit supports both interlaced (hardware-controlled) and non-interlaced (software-controlled) scanning.

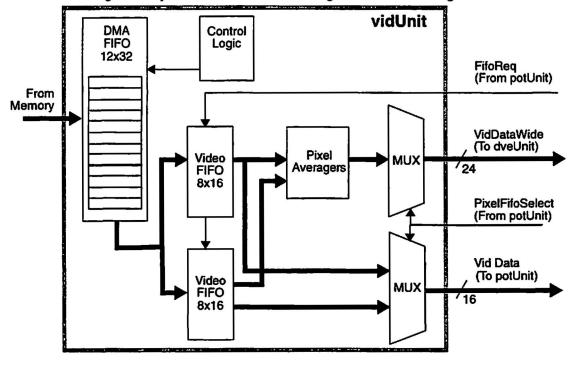


Figure 6-18 provides a functional block diagram of the vidUnit logic.

Figure 6-18 vidUnit Functional Block Diagram

6.9.2 Signal Definitions

The following table lists the vidUnit interface signals.

Table 6-9 vidUnit Interface Signals

Signal Name	Input/ Output	Signal Description	
vSync	I	Vertical sync	
hSync	Ī	Horizontal sync	
VidDataWide	I/O	Averaged pixel data	
vSync_sysclk	0	vSync aligned to the system clock	
hSync_sysclk	0	hSync aligned to the system clock	
HinActive	0	Starting horizontal address of active region on the screen	
VinActive	0	Starting vertical address of active region on the screen	
PixelFifoSelect	0	Selects vidUnit's video FIFOs	
FifoReq	0	Reloads vidUnit's DMA FIFO	

6.10 Digital Video Encoder Unit

6.10.1 Overview

The Digital Video Encoder Unit (dveUnit) is a Verilog RTL-level core. It converts RGB or YCC input pixels to either Composite Video/SVHS or Red/Green/Blue. The dveUnit generates either NTSC or PAL outputs at square or non-square pixel rates.

The dveUnit has two main modes of operation: YC and RGB. In YC mode, it generates Composite and SVHS signals compliant with the NTSC or PAL video standards. In RGB mode, it generates component Red, Green and Blue outputs with Sync inserted on the Green channel. The dveUnit accepts 8-bit digital linear RGB or CCIR601 YCbCr inputs. Various color space conversion modes are provided to match the input data to the required output format.

The data is filtered to limit the bandwidth of the signals to be within the supported ranges of the selected video standard. This filtering is performed using DSP techniques. The filters are programmable so that the dveUnit can either provide enhanced bandwidth for SVHS output, or correctly band-limited signals for Composite Video.

All the necessary synchronization signals for the NTSC and PAL standards are inserted into the composite and luma outputs. The correct sub-carrier burst frequency for color encoding is internally generated. Digital syncs are also provided for the rest of the system. Closed Captioning is available for both PAL and NTSC.

Two synchronous clocks are required for the dveUnit core, one at twice and the other at four times the pixel rate. The square pixel mode has a pixel rate of 12.2727...MHz for NTSC and 14.75MHz for PAL. The non-square pixel rate is 13.5Mhz for both NTSC and PAL. Most of the processing is performed at twice the pixel rate. The output rate is at four times the pixel rate, simplifying output filtering.

The dveUnit is a pure digital design. Three standard 9- or 10-bit DACs, and simple passive analog filters are needed, in addition to the circuitry provided. Eight-bit DACs can also be accommodated. During YC operation, the three DACs generate Composite, Luma and Chroma. During RGB operation, the DACs are used for Red, Green (with sync), and Blue.

Various trade-offs between functionality and size are available, and are described in more detail in the Verilog source and related documents.

Figure 6-19 provides a functional block diagram of the dveUnit logic.

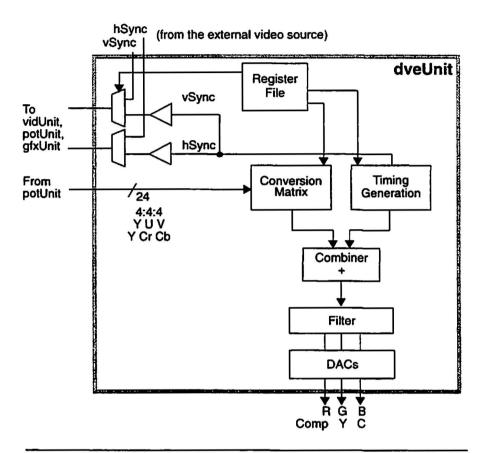


Figure 6-19 dveUnit Functional Block Diagram

6.10.2 Signal Definitions

The following table lists the dveUnit interface signals.

Table 6-10 dveUnit Interface Signals

Signal Name	Input/ Output	Signal Description
DAC_YVBS	I	VBS for the Y DAC
DAC_CRCBVBS	I	VBS for the CrCb DAC
DAC_COMPVBS	I	VBS for the Comp DAC
DAC_YAOUT	0	AOUT for the Y DAC
DAC_CRCBOUT	0	AOUT for the CrCb DAC
DAC_COMPAOUT	0	AOUT for the Comp DAC

Table 6-10 dveUnit Interface Signals (Continued)

Signal Name	Input/ Output	Signal Description
DAC_YAPVD	0	APVD for the Y DAC
DAC_CRCBAPVD	0	APVD for the CrCb DAC
DAC_COMPAPVD	0	APVD for the Comp DAC
DAC_YAPVS	0	APVS for the Y DAC
DAC_CRCBAPVS	0	APVS for the CrCb DAC
DAC_COMPAPVS	0	APVS for the Comp DAC
DAC_VREF	I	DAC voltage reference

6.11 Memory Unit

6.11.1 Overview

SOLO1's memUnit provides 32-byte burst read/write access to up to 64Mb of external SDRAM. The maximum achievable memory bandwidth is 333Mb/sec when 125MHz SDRAMs are used. The major functional blocks comprising the memUnit are:

- Control Registers
- Control Block
- Address Path/Arbiter
- Data Path

The data path is simply a multiplexer for write data, which is selected by the arbiter. The read data is registered into SOLO1 and broadcast to all read data recipients. Acknowledge signals for read and write data are generated separately for each requestor by the Control Register block.

The Control Register block shows the four memUnit programmable registers that maintain parameters that are used in the operation of the memUnit. Not shown in the Control Register block is a state machine which is responsible for initiating refresh.

The Address Path and Arbiter block contain the arbitration logic and the address selection logic as shown.

The timing of all requests, memory control signals and acknowledgments is from within the control block. The control is divided into 3 main units:

- Burst control
- Word Control
- Refresh Control

A fourth entity in the control block are the bank timers. These timers use the values programmed for the SDRAMs to control the sequencing and progress of the other control blocks.

The burst control is responsible for the timing of all burst reads and writes. Likewise, the word control is responsible for the gfxUnit word accesses, and the refresh control for the refresh accesses.

Figure 6-18 provides a functional block diagram of the memUnit logic.

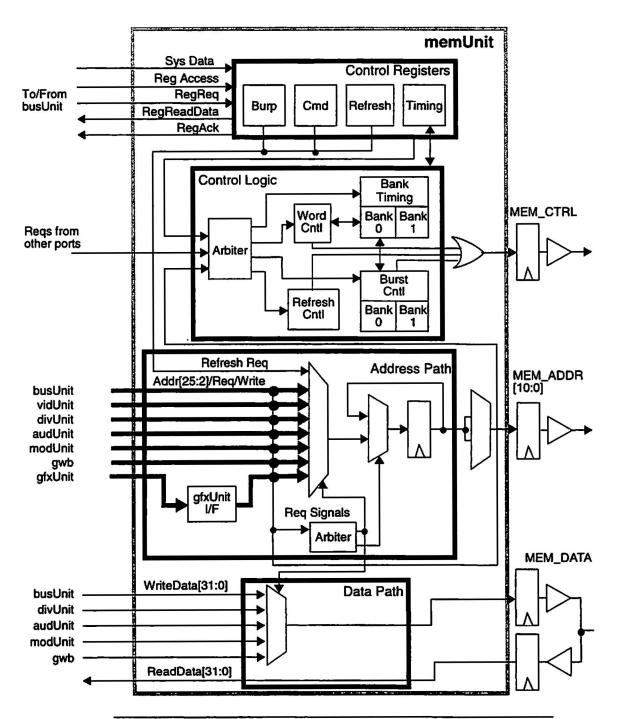


Figure 6-20 memUnit Functional Block Diagram

The internal ports that are serviced by the memUnit are listed in Table 6-11.

Table 6-11 memUnit's Supported Internal Ports

Unit	Access Type	Arbitration Type
Refresh	Refresh	Fixed: Top Priority
vidUnit (FB readout)	R Burst	Fixed: 1st Priority
divUnit	W Burst	Fixed: 2nd Priority
audUnit	R/W Burst	Fixed: 3rd Priority
modUnit	R/W Burst	Fixed: 4th Priority
busUnit (CPU interface)	R/W Burst	Round-robin
gwb (FB write-back)	W Burst	Round-robin
gfxUnit (graphics rendering)	Random	Round-robin, Top when active

6.11.1.1 Access Technique

SOLO1's high memory bandwidth is achieved through an efficient SDRAM access scheme. With the exception of the gfxUnit, all memory requests are 32-byte burst requests consisting of 8 consecutive cycles of 32 bits each. On power-up, software programs the SDRAM chips (via the memUnit) to provide or accept data in 4-cycle increments. Note however, that each SDRAM chip contains two independent banks, with each bank providing half of the requested data. For example, when the memUnit services a read request, it first accesses Bank 0 for four words, then Bank 1. The read from Bank 0 is called a "read with auto-precharge," meaning that once the fourth word is read from Bank 0, that bank will automatically precharge and be ready for a new access within four cycles (assuming the fastest SDRAM chips). Thus, the memUnit "ping-pongs", or pipelines the access to the SDRAMs and can keep data flowing at up to one word per cycle.

To accommodate slower, lower-cost SDRAMS, the timing parameters of the SDRAM chips can be programmed into the memUnit, so that the cost/performance trade-off can be made without compromising system cycle time or SDRAM timing specifications.

6.11.1.2 Arbitration Algorithm

The memUnit arbitrates between SOLO1's internal requestors, using a combination of fixed and round-robin schemes (refer to Table 6-11). Some requestors arbitrate with a fixed-request priority, while others, which don't require service in finite time, arbitrate after the fixed-priority requests in a round-robin scheme. This guarantees timely accesses to video and audio, as well as fair accesses to the busUnit, graphics write-back, and graphics rendering. Also, when graphics rendering (via the gfxUnit) has been arbitrated, it becomes the highest-priority requestor—it can't be kicked off until it's finished.

This arbitration allows the gfxUnit to take advantage of a random memory access mode (Word Mode) which only it can use. In Word Mode, the memUnit keeps pages open, and allows the gfxUnit to provide a new address for a new word within that page every cycle—Burst Mode is effectively defeated. This greatly improves the efficiency of the gfxUnit when rendering textures (particularly rotated textures), or any odd-stride memory access patterns. However, this efficiency would be lost if re-arbitration occurred for each word of access—necessitating the lock-out. To counteract the gfxUnit arbitration lock-out, a "burp" mechanism is used. This mechanism resets and enables a counter whenever the gfxUnit is accessing memory in the presence of a "burp-enabled" request. When the counter reaches a value determined by a CPU-programmed register, the gfxUnit loses the memUnit arbitration, and all other higher-priority requestors are serviced so that their real-time requirements are met. The burp terminal count and the burp enables are in the mem_BURP_Register.

6.11.1.3 Memory Refreshing

The memUnit is also responsible for memory refresh. The refresh interval is programmable, or refresh can be entirely disabled (i.e., the Power Up state). Unlike traditional DRAMS, the refresh address is maintained by the SDRAM chips themselves, so that the memUnit only needs to initiate refresh cycles.

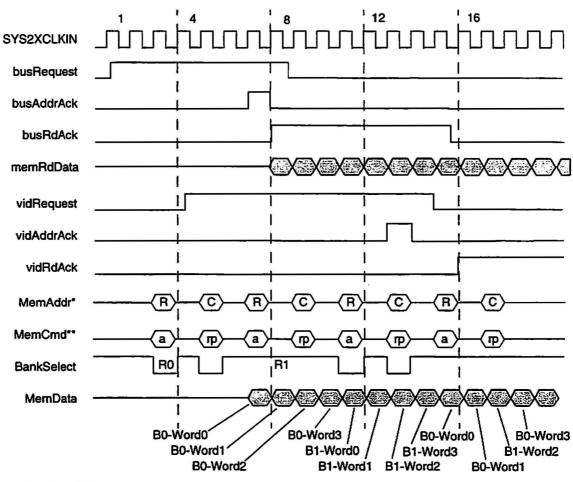


Caution: Auto-refresh mode works with all SDRAMs with SOLO1 Rev 1.3 and later.

Another feature of SDRAM is auto-refresh. The memUnit facilitates auto-refresh in the following way: If software wants to shut down most activity in the system to conserve power, then SDRAM accesses become unnecessary except for refresh. Software can program the SDRAMs to refresh themselves. It does so by setting the Auto-refresh enable bit in the MEM_REFRESH register. The memUnit will issue the command to the SDRAMs at the first refresh interval, where no requests have been made since the last refresh interval. Once the SDRAMs are in auto-refresh mode, any memory request will cause the memUnit to issue an auto-refresh exit command, so the request can be subsequently serviced.

6.11.1.4 memUnit Transaction Timing

Figure 6-21 illustrates two memory-read transactions performed back-to-back. The first read is with busUnit data, the second read is with vidUnit data.



^{*} R = Row Address

Figure 6-21 memUnit Transaction Timing

C = Column Address

^{**} a = Activate Row

rp = Read with Auto Precharge

6.11.2 Signal Definitions

The following table lists the memUnit interface signals that are external to SOLO1.

Table 6-12 memUnit Interface Signals

Signal Name	input/ Output	Signal Description
mem_SYS_2XCLKIN	I	2X system clock
MEM_ADDR[10:0]	0	Memory and address
MEM_DATA[31:0]	NO	Memory data bus
MEM_BS	0	Memory bank select
MEM_DQM[3:0]	0	Byte enables
MEM_RAS_N	0	Row address strobe
MEM_CAS_N	0	Column address strobe
MEM_WE_N	0	Write enable
MEM_CS_N[1:0]	0	Chip selects
MEM_CKE	0	Clock enable

6.12 Smart Card and UART Control Unit

6.12.1 Overview

The Smart Card and UART Control Unit (sucUnit) provides the interfaces to two Universal Asynchronous Receiver/Transmitter (UART) devices. Each one of these UART devices may be configured to act as a Smart Card interface, a UART with a modem interface, or a set of general-purpose inputs/outputs (GPIOs) under software control. The expected configuration is to implement one Smart Card reader and one general-purpose UART.

A block diagram of the sucUnit is shown in Figure 6-22. It consists of an interface to the busUnit (sucBusIF), the two configured UARTs (sucUart) and some logic to MUX the internal state for diagnostics (sucDiag) onto some of the output pins. The two sucUart modules are referred to as SC0 (Smart Card 0) and GPU (general-purpose UART) to reflect how they will most likely be used.

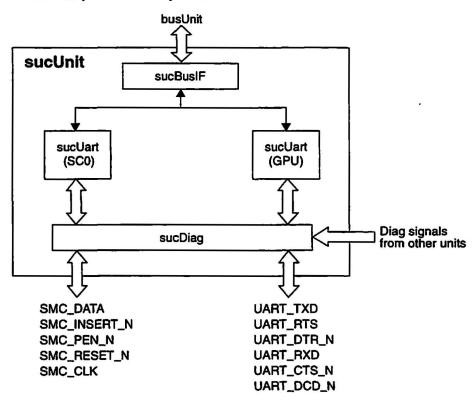


Figure 6-22 sucUnit Functional Block Diagram

When one of the interfaces is configured as a Smart Card, external logic is required in order to correctly implement an ISO-7816 or EMV (Europay/MasterCard/Visa) compliant interface. The external logic must include a 5V power enable circuit (to supply power to the card under software control), over-current sensing logic, and clock transition logic.

6.12.2 Signal Definitions

The sucUnit signals are described in Table 6-13.

Table 6-13sucUnit Signals

Signal Name	Directio n	Description
SMC_DATA	ΙΟ	Smart Card data
SMC_INSERT_N	ΙΛΟ	Smart Card inserted
SMC_PEN_N	I/O	Smart Card power enable
SMC_RESET_N	I/O	Smart Card reset
SMC_CLK	1/O	Smart Card clock
UART_CTS_N	ΝΟ	UART clear to send
UART_DCD_N	ΝO	UART data carrier detect
UART_DTR_N	I/O	UART data terminal ready
UART_RTS_N	I/O	UART request to send
UART_RXD	ΙΛΟ	UART receive data
UART_TXD	ΙΛΟ	UART transmit data

6.12.3 Submodule Descriptions

6.12.3.1 sucUnit Bus Interface Submodule (sucBusIF)

The sucBusIF submodule handles the WebTV Intermodule Interface (WTI) to the busUnit. It only implements register reads and writes to the sucUnit. There is no support for any direct memory access. All software-accessible registers are in the two sucUarts. The sucBusIF does the first-level decode of the sucUnit address space and asserts the proper read/write strobes and control signals to the two sucUart submodules. It is responsible for issuing transaction acknowledgments and read data from the sucUart submodules to the busUnit. All outputs of the sucUnit are driven directly from registers.

6.12.3.2 sucUnit UART Submodule (sucUart)

Each sucUart submodule implements two UART interfaces—Smart Card and serial/modem port. In addition, it implements a GPIO interface that can be enabled on a per-pin basis. If a pin is enabled to be a GPIO, that will override any Smart Card or serial port function of that pin. All configuration is under software control.

The pin mapping supporting the two modes is shown in Table 6-14. Note that the Smart Card interfaces (SCO and SC1) have only five pins, so the UART_DCD_N and GPIO[5] outputs are not available on the Smart Card ports.

Table 6-14 sucUart Interface Modes

Serial/Modem Mode	Smart Card Mode	GPIO Pin	
UART_RXD	SMCn_DATA	GPIO[0]	
UART_TXD	SMCn_CLK	GPIO[1]	
UART_DTR_N	SMCn_PEN	GPIO[2]	
UART_RTS_N	SMCn_RESET	GPIO[3]	
UART_CTS_N	SMCn_INSERT	GPIO[4]	
UART_DCD_N	** NA **	GPIO[5]	

A block diagram of the sucUart submodule is shown in Figure 6-23. The sucURegs module contains all of the software-accessible registers for controlling and configuring the UARTs. Each UART contains independent receive and transmit FIFOs, as well as transmit and receive shift registers. The receive and transmit FIFOs are swappable under software control. There is also a GPIO interface on each UART, so that each pin can be configured as a general-purpose output or input. Each GPIO is capable of generating interrupts on either rising transitions, falling transitions or both.

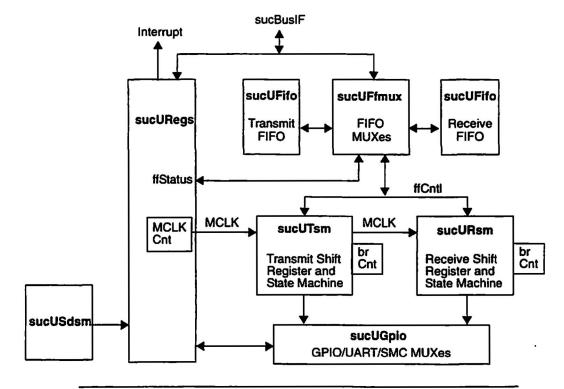


Figure 6-23 sucUart Submodule Functional Block Diagram

The transmit FIFO is used to store transmit data until it can be shifted out serially. The receive FIFO is used to store receive data until software can read it. To a large degree, the depth of the FIFOs determine how fast the UART can run, so that software can keep up with the data flow. Deeper FIFOs enable software to reduce the time between servicing interrupts.

Aside from the depth of the FIFOs, the SCO and GPU UARTs are functionally identical. It is up to software to properly configure each of the UARTs for the intended operation.

The SCO UART has 1-byte receive and transmit FIFOs (1T/1R). The GPU UART has a 4-byte receive FIFO and an 1-byte transmit FIFO. These FIFOs can be swapped so that the UART is configured with 1-byte receive/4-byte transmit FIFOs. The GPU should be capable of 115.2Kbps, depending on the system software. SCO should be capable of at least 19.2Kbps.

The sucUnit UARTs have many features of the industry-standard 16550 UART. Although the register map and bit definitions are different from the 16550, the sucUnit UARTs should implement the features of the 16550 required for software.

This documentation will point out the similarities and differences between the sucUnit UARTs and a 16550 UART.

The clock rate for the sucUnit UARTs is derived from the system clock, which may range from 66MHz to 83MHz. Software is responsible for programming divisors from the system clock to generate the UART master clocks and the bit-rate clocks.

6.12.3.3 sucUnit Diagnostic Module (sucDiag)

The sucDiag submodule is used to override six of the sucUnit outputs with the internal state of the chip. This is used as a debug tool to gain visibility into the internal state machines in various units. The values and enables for the diagbus are controlled by registers in the devUnit. If the diagbus is enabled for output, it will override all normal operation of the output signals.

The diagbus is connected to the chip outputs as shown in Table 6-15.

sucUnit Pin	diagbus Pin
SMC1_DATA	dev_diagBusOut[0]
SMC1_INSERT	dev_diagBusOut[1]
SMC1_CLK	dev_diagBusOut[2]
SMC1_RESET	dev_diagBusOut[3]

Table 6-15 diagbus Output Pins

SMC1_PEN

UART_DTR_N

6.12.4 WebTV Terminal EMV Specification Compliance

dev_diagBusOut[4]

dev_diagBusOut[5]

There are two standards describing Smart Card protocols and specifications: the ISO-7816 standard and the EMV (Europay/MasterCard/Visa) Specification 1996. The EMV specification is a restricted subset of the ISO-7816 specification. The sucUnit Smart Card interface attempts to be compliant with the EMV specification. This section can be used as a standalone document for reviewing how well the WebTV terminal complies with the EMV specification.

The following sections address how the sucUnit, in a Smart Card configuration, complies with the specifications set forth in the "EMV '96 Integrated Circuit Card Specification for Payment Systems, Part I." These sections state how the WebTV Terminal implements the various EMV features on a section-by-section basis that matches the sections in the EMV '96 specification (the EMV section numbers are listed in parentheses next to the corresponding section headings).

6.12.4.1 EMV Compliance Overview

WebTV has designed a device that can be used as a Smart Card Terminal (WebTV Terminal) with two Smart Card interfaces. The WebTV Terminal has a custom ASIC (WebTV ASIC) to implement the two Smart Card interfaces. This ASIC is controlled by an on-board CPU. The two Smart Card interfaces on the ASIC will interface directly to Smart Cards if possible. The interfaces support either synchronous ISO-7816 compliant cards or asynchronous EMV-compliant cards.

The Smart Card clocks are generated by dividing down the 83-MHz system clock frequency in the WebTV ASIC.

Each Smart Card interface on the ASIC supports the following signals:

SMC_DATA	(the I/O signal in the EMV spec)
SMC_INSERT_N	(insert/removal detect from a switch)
SMC_PEN_N	(power enable, controls power on/off to card)
SMC_RESET_N	(the RST signal in the EMV spec)
SMC_CLK	(the CLK signal in the EMV spec)

All of these signals are controllable by software in terms of their direction and the values that they can drive when enabled as outputs. Additionally, any one of these signals may be configured to interrupt the CPU on a rising or falling transition. Some of these pins (i.e., SMC_DATA and SMC_CLK) may have special functions, such as UART serial transmission or clock generation, which are enabled by software, but are under hardware control. This specification attempts to explicitly distinguish which functions are entirely under software control and which ones have hardware support.

6.12.5 Electromechanical Interface (EMV '96 Part I, 1)

6.12.5.1 Electrical Characteristics of the Terminal (EMV '96 Part I, 1.4)

6.12.5.1.1 Input/Output (I/O) (EMV '96 Part I, 1.4.2)

From the EMV '96 spec: "During operation, the terminal and the ICC shall not both be in transmit mode. In the event that this condition occurs, the state (voltage level) of the contact is indeterminate and no damage shall occur to the terminal." Software will ensure that the ICC and terminal are not in transmit mode at the same time.

The open-drain devices driving the data signal guarantee that no damage will occur to the terminal or the ICC if both are in transmit mode. If the Philips 8002 interface chip is used, it may contain circuitry to ensure this. If the Philips 8002 is not used, the I/O pin may be implemented as an open-drain signal so that if there are any drivers on the signal, they will only drive low.

The terminal will pull the I/O signals to the switched VCC to ensure that if the terminal and the ICC are both in reception mode, the ICC contact will be in the high state. The EMV '96 spec states: "The terminal shall not pull I/O high unless VCC is powered and stable within the tolerances specified in Section I-2.1.2."

The I/O signal will be pulled high with a passive resistor connecting the I/O signal to the switched VCC. Software will drive the I/O signal low before enabling power to VCC. After a fixed duration, but before driving CLK high, software will tri-state the I/O signal and it will be pulled to VCC.

6.12.5.1.2 Transmission Mode (EMV '96 Part I, 1.4.2.1)

The I/O signal will meet the specifications for Voh, Vol, tR, tF, overshoot and undershoot specified in Section 1.4.2.1 of the EMV `96 specification. In transmission mode, the WebTV terminal will drive the I/O signal to either the high or low state.

6.12.5.1.3 Reception Mode (EMV '96 Part I, 1.4.2.2)

The WebTV terminal will operate correctly under the Vih, Vil, tR and tF specified for I/O in Section 1.4.2.2 of the EMV '96 specification. In reception mode, the WebTV terminal will tri-state the I/O signal and monitor it for incoming data from the ICC.

6.12.5.1.4 Programming Voltage (VPP) (EMV '96 Part I, 1.4.3)

The WebTV terminal will not supply VPP.

6.12.5.1.5 Clock (CLK) (EMV '96 Part I, 1.4.4)

The CLK signal will meet the specifications for Voh, Vol, tR, tF, overshoot and undershoot specified in Section 1.4.4 of the EMV '96 specification.

The clock will be divided down from the WebTV terminal system frequency, which will be between 66MHz and 83MHz. The exact divisor will be under software control and will be set to produce a clock in the 1-5MHz range. The clock high and clock low time will be the exact same number of cycles, so that the only duty-cycle implications will be from the rise and fall times of the buffers. The duty cycle will be well within the 45/55% duty cycle.

6.12.5.1.6 Reset (RST) (EMV '96 Part I, 1.4.5)

The CLK signal will meet the specifications for Voh, Vol, tR, tF, overshoot and undershoot specified in Section 1.4.5 of the EMV '96 specification.

6.12.5.1.7 Supply Voltage (VCC) (EMV '96 Part I, 1.4.6)

The exact voltage range and steady state output current of VCC is less than 700mA. The WebTV terminal has protection circuitry to prevent short circuits to VCC or GND. Transients in current consumption greater than 200mA will deactivate the ICC.

6.12.5.1.8 Contact Resistance (EMV '96 Part I, 1.4.7)

The contact resistance is system dependent.

6.12.5.1.9 Short Circuit Resilience (EMV '96 Part I, 1.4.8)

The WebTV terminal is capable of sustaining a short circuit of any duration between any or all contacts. Any short circuit will result in deactivation of the ICC.

6.12.5.1.10 Powering and Depowering of Terminal with ICC in Place (EMV '96 Part I, 1.4.9)

If the WebTV terminal is powered on or off with an ICC in place, no spurious signals or power perturbations shall appear at the interface contacts. In the event of unexpected removal of power from the WebTV terminal, the ICC will be deactivated before the contacts have disengaged. When power is applied to the WebTV terminal, no signals will be driven to the ICC. The I/O (SMC_DATA), RST (SMC_RESET), and CLK (SMC_CLK) signals will not be driven and pulled to an inactive state with resistors. The switched VCC to the ICC will be disabled until software enables it (through SMC_PEN) and goes through the correct activation sequence. The card will be detected if inserted by software (through SMC_INSERT) after power has been applied.

6.12.6 Card Session (EMV '96 Part I, 2)

6.12.6.1 Normal Card Session (EMV '96 Part I, 2.1)

6.12.6.1.1 Stages of a Card Session (EMV '96 Part I, 2.1.1)

6.12.6.1.2 ICC Insertion and Contact Activation Sequence (EMV '96 Part I, 2.1.2)

On insertion of the ICC into the WebTV terminal, the I/O, RST, and CLK contacts of the ICC will be either driven low or pulled low with passive resistors. The VCC will not be switched on and will be below 0.4V.

The insertion of the ICC into the WebTV terminal will be detected through a mechanical switch pulled up to the unswitched main power supply on the WebTV terminal. Once insertion is detected, software will initiate the activation sequence.

The activation sequence is as follows:

- 1. The RST, CLK and I/O signals are driven low.
- 2. VCC is then powered. After a fixed duration under software control, VCC will be assumed to be stable.

- 3. I/O will be put into reception mode by tri-stating the signal.
- 4. The WebTV terminal will then go through the reset sequence for a synchronous ICC. If the ICC is not detected as a synchronous card, the reset sequence specified for an asynchronous EMV ICC will be applied.

6.12.6.1.3 ICC Reset (EMV '96 Part I, 2.1.3)

The WebTV terminal supports both synchronous and asynchronous ICCs. The reset for synchronous ICCs will be applied first. If the Answer To Reset (ATR) indicates the card is a synchronous card, no further reset sequences will be issued to the ICC and it will be assumed to be a synchronous device. If the ATR does not indicate a synchronous ICC, the reset sequence for an asynchronous ICC will be applied. If the ATR does not indicate an asynchronous ICC, the card will be rejected and the deactivation sequence will be applied.

The reset sequence for an asynchronous ICC is specified in the "Cold Reset" and "Warm Reset" sections that follow.

6.12.6.1.4 Cold Reset (EMV '96 Part I, 2.1.3.1)*

Following the activation of contacts (see "ICC Insertion and Contact Activation Sequence" on page 56), the WebTV terminal will initiate a cold reset as follows:

- A 1-5MHz clock is applied to the CLK signal at notational time T0.
- 2. The clock is generated from the WebTV terminal system clock as stated in "Clock (CLK)" on page 55.
- 3. The RST signal remains in the low state for a period of between 40000 and 45000 clock cycles after time T0. At this point (time T1), the RST signal is put in the high state. RST is implemented as an open-drain output pulled up to the switched 5V power supply. Software is responsible for waiting the appropriate amount of time and setting RST high between 40000 and 45000 cycles after time T0.
- 4. After RST is set high at time T1, the WebTV terminal waits to receive data from the ICC. If software does not detect any data for a period of time greater than 40000 cycles, it initiates the deactivation sequence specified in the "Contact Deactivation Sequence" section. If the WebTV terminal detects data from the ICC and it conforms to the expected ATR, the card is treated as a valid card and the transaction execution begins. If the data does not conform to the expected ATR, a warm reset will be issued, as described in the following section.

6.12.6.1.5 Warm Reset EMV '96 Part I, 2.1.3.2)

If the ICC issued data in response to the cold reset sequence, but it did not conform to the expected ATR, the WebTV terminal will initiate a warm reset as follows:

 The WebTV terminal keeps the CLK signal running according to Section 1.4.4 of the EMV '96 specification.

- VCC will remain stable and within the specifications (see Section 1.4.6 of the EMV '96 specification).
- 3. At notational time T0, the WebTV terminal will set RST to the low state. The I/O signal will be set to reception mode prior to setting RST at time T0.
- 4. The rest of the warm reset will be identical to the cold reset from time T0.
- 5. If the ICC does not respond with the answer to reset within 40000 cycles of T1 or if it responds with an unexpected ATR, the ICC will be rejected and the WebTV terminal will start the process specified in the "Contact Deactivation Sequence (EMV '96 Part I, 2.1.5)" section that follows.

6.12.6.1.6 Execution of a Transaction EMV '96 Part I, 2.1.4)

Once the ICC is determined to be valid, all subsequent information exchanged between the ICC and the WebTV terminal will be under control of the terminal's application software.

6.12.6.1.7 Contact Deactivation Sequence (EMV '96 Part I, 2.1.5)

The contact deactivation sequence will be started, upon either normal or abnormal termination (including withdrawing the ICC while being accessed by the WebTV terminal). This sequence is handled entirely in hardware, and will be initiated either by the detection of card removal or by software initiating the deactivation sequence.

The WebTV terminal will set the the RST signal to the low state. The CLK signal will remain active for approximately 10us after the RST signal is set low. At this time, CLK will be driven low by the WebTV terminal. The CLK duty cycle will not be violated in setting the CLK signal low. Approximately 10us after setting CLK low, the I/O line will be set to a high impedence state.

After setting both the RST and CLK signals low, and I/O to the high impedence state, the WebTV terminal will depower VCC to the ICC. VCC will be depowered approximately 10us after setting I/O to a high impedence state. It will take a maximum of 500msec for VCC to drop below 0.4V. VCC will be depowered before the ICC contacts are physically disconnected from the WebTV terminal.

If the ICC is unexpectedly removed, the WebTV ASIC will detect the removal via an "insert" signal from the card socket. This signal will be active when the card is inserted and will deactivate when the physical removal of the card begins. The deactivation process will complete before the contacts disengage as long as the ICC is removed at the rate of 1m/sec or less. Removal of the card will force the deactivation sequence to occur without any software interaction.

6.12.6.1.8 Abnormal Termination of Transaction Process (EMV '96 Part I, 2.2)

If an ICC is prematurely removed at speeds of up to 1 m/s, during the execution of a transaction, the WevTV terminal will sense the movement of the contacts and deactivate all signals to the contacts within 1 ms. This ensures that all contacts will be deactivated before the contact movement exceeds 1 mm. This will not cause any electrical or mechanical damage to the ICC.

6.12.6.2 Physical Transportaion of Characters (EMV '96 Part I, 3)

6.12.6.2.1 Bit Duration (EMV '96 Part I, 3.1)

The bit duration on the I/O line is defined in terms of elementary time units (ETUs). The definition of an ETU is:

where:

f = the CLK frequency in Hz

F = the clock rate conversion factor

D = the bit rate adjustment factor

The WebTV terminal will support programmable integer F/D ratios. Software will program a 13-bit divisor of the system clock to generate the sample frequency. The sample frequency will be 16 times the bit rate (1/ETU).

The divider used to generate the sample clock, along with the 6-bit divider used to generate CLK, give the WebTV terminal flexibility in generating various F/D ratios. Assuming F = 372, the WebTV terminal will support values for D of 1/16, 1/8, 1/4, 1/2, 1, 2, 4, 8, and 16, as specified by the "ISO/IEC 7816-3 Specification."

Note: For the answer-to-reset protocol, the F/D ratio will be set to 372.

6.12.6.2.2 Character Frame (EMV '96 Part I, 3.2)

The WebTV terminal will implement a standard UART interface to receive and send data over the I/O line. The line will have the following characteristics:

- Prior to character transmission, the I/O line will be in state H.
- A character will consist of 10 bits, including one start bit (state L), 8 data bits, and one
 parity bit.
- Even parity will be used. The parity will be set to make the total number of logical '1's (the 8 bits of data plus the parity bit) even.
- A programmable number of stop bits (state H) ranging from 1 to 256.

The sample time is fixed to be 1/16 ETU. The start bit is recognized by detecting a falling edge using CLK as the clock for the edge detection. The state of the bit is determined by sampling n+0.5 ETUs after the detection of the falling edge of the start bit (where n is the nth bit in the character).

The guard time is implemented as a programmable number of bits between 1 and 256. If the WebTV terminal was transmitting, it will not drive the I/O line after the parity bit. The on-board passive pullup resistor will pull the I/O bit high within 1 us. If the WebTV terminal was receiving, it will stay in receive mode for the duration of the guard time.

If the card is determined to support T=0 (character-oriented asynchronous transmission protocol), the WebTV terminal will implement a parity error notification scheme. If the terminal is transmitting, it will sense if a parity error was detected by the receiver. If the I/O line is low at 11 ETUs after the leading edge of the character was sent, the terminal will issue an interrupt and software will send the character again. Software will transmit the character three more times if parity errors continue to be detected.

If the terminal is receiving a character and it detects a parity error (after the answer to reset), it will drive the I/O line low starting at 10.5 ETUs after the falling edge of the start bit for a duration of 1 ETU. Software will be notified of the parity error when it reads the receive data.

Software will guarantee that at the terminal transport layer (TTL), data will always be passed over the I/O line most significant byte first.

The order of the bit within the byte is configurable in hardware. Software will set it in the proper mode, based on the initial character in the answer-to-reset sequence (see the "Character Definitions" section, later in this chapter, for a description of the initial character).

6.12.6.3 Answer-To-Reset (EMV '96 Part I, 4)

6.12.6.3.1 Physical Transport of Characters Returned at Answer-to-Reset (EMV '96 Part I, 4.1)

During the answer-to-reset sequence, the WebTV terminal hardware does not impose any restrictions on the ICC in terms of maximum character intervals or total answer-to-reset times. It is up to software to determine if the reset sequence times out.

Parity error handling is controlled entirely by software. The hardware will be in receive mode for the entire duration of the answer-to-reset sequence.

6.12.6.3.2 Characters Returned by ICC at Answer-to-Reset (EMV '96 Part I, 4.2)

Handling the answer-to-reset sequence is controlled entirely by software. Both ICCs supporting T=0 (character-oriented asynchronous transmission protocol) and T=1 (block-oriented asynchronous transmission protocol) are supported by the WebTV terminal.

6.12.6.3.3 Character Definitions (EMV '96 Part I, 4.3)

TS - Initial Character (EMV '96 Part I, 4.3.1)

The TS initial character indicates the logic convention for subsequent characters and is used to facilitate bit synchronization.

The WebTV terminal does not use the initial TS character to do any bit synchronization. The initial character is detected just as any other: by the falling edge of the I/O line, which indicates the start bit.

The value of TS will be one of the following two values:

- (H)LHHLLLLLH (inverse convention, value 0x3F)
- (H)LHHLHHHLLH (direct convention, value 0x3B)

In the *inverse* logic convention, a low (state L) on the I/O line is equivalent to a logical one (1), and the most significant bit of the data byte is the first bit sent after the start bit.

In the *direct* logic convention, a high (state H) on the I/O line is equivalent to a logical one (1), and the least significant bit of the data byte is the first bit sent after the start bit.

The WebTV terminal supports both logic conventions in hardware. The initial word will be read using the direct logic convention. If the value is read as 0x3B with no parity error, it indicates that the direct logic convention will be used. If it is read as 0x03 with a parity error (the inverse convention 0x3F read as direct convention), the hardware will be programmed to use the inverse logic convention. The hardware will inverse the sense of the bits, as well as reorder the bits, so that software will always read the data byte in the direct logic convention.

Any card that does not read either the inverse or direct convention values will be rejected. The WebTV terminal will issue a warm reset, go through the answer-to-reset sequence, and then deactiveate the card if it still doesn't return one of the two expected values.

TO - Format Character (EMV '96 Part I, 4.3.2)

The T0 character is handled entirely in software.

TA1 to TC3 - Interface Characters (EMV '96 Part I, 4.3.3)

Only the characters that have hardware support are listed. The other characters (TD1, TA2, TB2, TC2, TD2, TA3, TB3 and TC3) are implemented entirely in software and are not specified here.

TA1 (EMV '96 Part I, 4.3.3.1)

The WebTV terminal is capable of reading the TA1 character, which supplies the values for F and D specified in the "Character Frame" section. If the values for F and D correspond to the ranges specified (F=371, D=1/16...16), the card will be accepted. Otherwise, it will be rejected. If TA1 is not specified, F and D will be set to '1.'

TB1 (EMV '96 Part I, 4.3.3.1)

The WebTV terminal does not supply VPP. It always assumes that the card returned TB1=0x00.

TC1 (EMV '96 Part I, 4.3.3.1)

TC1 specifies the extra guard time that shall be added to the minimum duration between the leading edges of the start bits, when the terminal is transmitting data.

The WebTV ASIC supports between 1 and 256 stop bits after the parity bit of the character was sent. Software must determine the correct number of stop bits to program, depending on the mode and the value of TC1. The time between the leading edges of characters will be somewhere between 11 and 256 ETUs.

TCK - Check Character (EMV '96 Part I, 4.3.4)

The "checksumming" of the answer-to-reset will be handled entirely in software.

6.12.6.3.4 Sequence and Conformance of Answer-to-Reset (EMV '96 Part I, 4.4)

The WebTV terminal's response to the ICC answer-to-reset is handled entirely in software.

6.12.6.3.5 Answer-to-Reset---Flow at the Terminal (EMV '96 Part I, 4.5)

No additional hardware implications.

6.12.6.4 Transmission Protocols (EMV '96 Part I, 5)

This section primarily describes the higher-level software protocols. It is assumed that software will be able to correctly implement these protocols without any additional hardware support, other than that specified in the previous sections. Areas having potential hardware implications are described in the following subsections.

Section 5.2.2.1, of the EMV specification, specifies that the minimum interval between the leading edges of the start bit or two consecutive characters sent in opposite directions shall be 16 ETUs. There will be no hardware support for this. Since software is responsible for turning around the direction of I/O data, it will also guarantee that 16 ETUs have elapsed before the WebTV terminal sends characters to the ICC.

The "Character Frame" section specifies the error detection and correction scheme for T=0. This is handled in hardware as specified previously. Basically, the terminal will notify the card of a parity error by a setting the I/O line to state L at 10.5 ETUs for a duration of 1 ETU. After transmitting a character, the terminal will see if the ICC detected a parity error by checking for state L on the I/O line at 11 ETUs after the line's leading edge. The software is responsible for re-issuing the disputed character.

6.13 Graphics Unit

6.13.1 Overview

The Graphics Unit (gfxUnit) is composed of the following logic blocks:

- Control Subunit (fctlUnit)
- Mapping Subunit (fmapUnit)
- Vector Quantitization Subunit (fvqUnit)
- Blend Subunit (fbldUnit)
- Ram Line Buffer Subunit (frlbUnit)
- Video Out Subunit (fvoUnit)
- Write Back Subunit (fwbUnit)

The fctlUnit interfaces with the busUnit for register reads and writes. It also interfaces with the memUnit in order to read yMaps, CelRecords, textures, and Codebooks.

The fctlUnit sends the information necessary to process each cel to each block of the gfxUnit data pipeline. The fmapUnit takes all of the celrecord parameters relating to the cel's shape and position on the screen and creates (u,v) values for each pixel. The (u,v) values are used to calculate the memory address in the texture for each pixel to be displayed.

The fvqUnit takes the data words read from memory and according to the pixel mode (Dir444, Dir422, VQ8 444, VQ8 422, VQ4 444) presents the Y, Cb, Cr values (luminence and chrominance) to the fbldUnit.

The foldUnit then performs a read-modify-write operation on the internal line buffers to blend the new foreground pixel into the background already stored in the line buffers (frlbUnit).

If in graphics ping-pong mode, these line buffers are read out in real time and sent to the potUnit for display.

If in graphics Write-back mode, each line buffer is read out once it has finished being composed. The fwbUnit sends each line to the proper place in the write-back frame buffer located in memory.

Figure 6-24 provides a functional block diagram of the graphics engine logic.

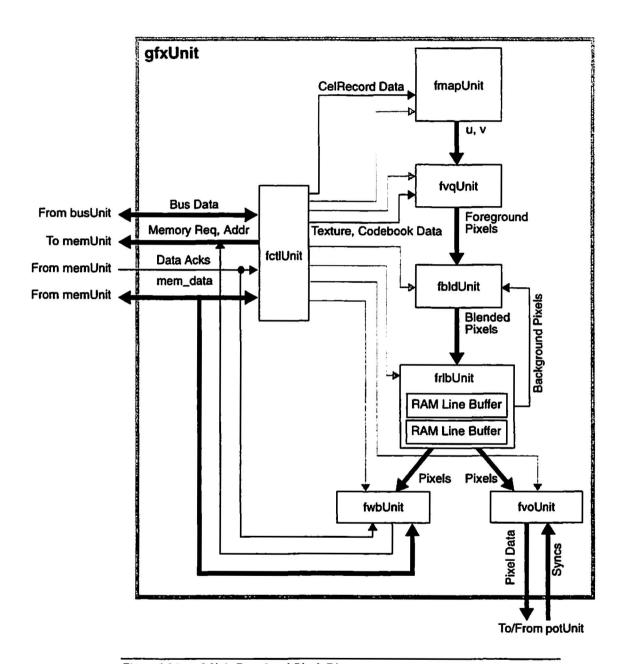


Figure 6-24 gfxUnit Functional Block Diagram

6.13.2 Signal Definitions

The following table lists the gfxUnit interface signals.

Table 6-16 gfxUnit Interface Signals

Signal Name	Input/ Output	Signal Description
io_SYS_2XCLKIN	I	System clock
io_POT_CLK	I	2X pixel clock
disp_CbCrSel	Ī	Selects either Y or Cb/Cr as current data
Interface to mckUnit		
mck_fbldUnit_areset_n	Ī	Asynchronous reset
mck_fctlUnit_areset_n	I	Asynchronous reset
mck_fmapUnit_areset_n	I	Asynchronous reset
mck_fvqUnit_areset_n	I	Asynchronous reset
mck_fwbUnit_areset_n	I	Asynchronous reset
ckd_gfxUnit_aresetPot_n	I	Reset synchronized to POT_CLK
Interface to potUnit		
pot_hsync	I	Synchronized version of the horizontal sync signal
pot_vsync	I	Synchronized version of the vertical sync signal
vid_earlyHblank	I	Early version of horizontal blanking signal
vid_oddField	I	Indicates an odd or even field
gfx_pixelBus [15:0]	0	Pixel bus to potUnit
gfx_pix444 [23:0]	0	Pixel bus to dveUnit
Interface to memUnit		
mem_gfxAddrAck	I	Address acknowledge
mem_gfxRdAck	I	Read acknowledge
mem_data [31:0]	I	Data bus from memUnit
gfx_addr [25:2]	0	Current address to memUnit
gfx_addrNext [25:2]	0	Next address to memUnit
gfx_memReg	0	Memory request
gfx_memReqNext	0	Next memory request
mem_gwbAddrAck	1	Write-back address acknowledge

Table 6-16 gfxUnit Interface Signals (Continued)

Signal Name	Input/ Output	Signal Description
mem_gwbWrAck	I	Write-back write acknowledge
gwb_data [31:0]	0	Write-back data to memUnit
gwb_addr [25:2]	0	Write-back address to memUnit
gwb_memDqm [3:0]	0	Byte enables
gwb_memReq	0	Write-back memory request
Interface to busUnit		
bus_gfxReq	I	Request from busUnit
bus_addr [31:0]	I	Address from busUnit
bus_data [31:0]	I	Data from busUnit
bus_write	I	Write signal from busUnit
gfx_data [31:0]	0	Data to busUnit
gfx_busAck	0	Acknowledge signal to busUnit
gfx_Int	0	Interrupt to busUnit
Interface to devUnit		
dev_ramDelay [7:0]	I	Control signals that set programmable de- lay in the RAM enable path
gfx_diagBus [5:0]	0	Output signals to debug bus
Test Signals		
test_ramAddr [8:0]	1	RAM test address
test_ramEnCycle [3:0]	I	RAM test enable
test_ramTestRamSel [3:0]	I	RAM test—select this RAM
test_ramRw	I	RAM test read/write
test_ramDataIn [23:0]	I	RAM test data input
test_ramDataOut [23:0]	0	RAM test data output
test_ramTestMode	I	RAM test mode
test_ramSelPortA	I	RAM test—select Port A

6.13.3 Control Subunit

The control logic consists of a State Machine and the address generation logic.

6.13.3.1 State Machine

The gfxUnit State Machine is initialized at the beginning of HSYNC (or the alternate start-of-line signal if in Write-back mode). The State Machine initializes the yMapBase and celsBase registers by copying over the yMapBaseMaster and the celsBaseMaster. The gfxUnit then reads a single yMap entry. Each yMap entry is exactly one word and refers to one CelBlock. The topLine and Height parameters are compared to the current line count (yCount). The CelBlock is active if (yCount < bottomLine) & (yCount >= topLine). If the CelBlock is not active, FIDO_GRFX proceeds to read another yMap entry.

If the CelBlock is active, the celRecord parameters are loaded into the graphics engine. First, the internal parameter registers are initialized with default values. Then, the graphics engine uses the CelSize bits of the yMap entry to read in the appropriate number of words. A microCelRecord is two words, a miniCelRecord is four words, and a full CelRecord is 12 words. The CelRecord parameters overwrites the default values. For the microCelRecord and the miniCelRecord, only a portion of the parameters are overwritten and the rest of the registers retain the default values.

Once the CelRecords parameters are loaded, the gfxUnit decodes the celMode byte to find out what type of Cel it is. If it is a LoadData celRecord, the Graphics Engine proceeds to read in the appropriate data and then moves on to the next yMap entry. Otherwise, the data pipeline is used to prepare and write image data into the scanline buffer.

The data pipeline includes: performing the optional transform operation, reading the texture from memory, performing the optional VQ decompression and blending the resulting pixel into the frame buffer.

When all the pixels have been processed for that line of the Cel, processing moves on to the next Cel in the CelBlock. When the last Cel in the CelBlock has been processed, the State Machine goes back to reading yMap entries to find the next active CelBlock.

During a ping-pong operation, it is possible that an image could be presented to the Graphics Engine that is too complex to process in one scanline time. If an HSYNC pulse arrives when the State Machine is NOT in Idle mode, then the current line is aborted, an interrupt is generated, and processing moves on to the next line.

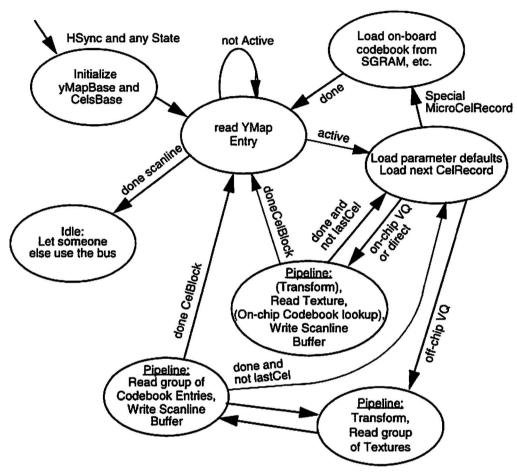


Figure 6-25 State Machine Diagram

6.13.3.2 Address Generation

The Control subUnit computes the address in memory for all the different types of memory reads. This section provides descriptions of each of the supported addressing modes.

6.13.3.2.1 Direct Map 4:4:4 Addressing For Direct Map 4:4:4 address calculation, a horizontal and vertical address is calculated for each pixel fetched. The vertical address is multiplied times rowBytes, it is added to the horizontal address, and this sum is added to the textureBase address. This address is presented to the memory bus interface, and a 4:4:4 pixel is fetched. Note that the memory system is 32 bits wide, so the address generated is long-word aligned. Specifically:

pixelAddr = (verticalAddr * textureRowLongs + horizontalAddr) * 4 +
textureBase

6.13.3.2.2 Direct Map 4:2:2 Addressing Direct Map 4:2:2 address calculation is similar to 4:4:4 Addressing, but since there are two pixels in each 32-bit word, the LSB of the horizontal address is saved, and then the calculated address, plus the horizontal address divided by 2, is presented to the memory system. The saved LSB is then used as a selector to determine which Y is to used from the YCbYCr pixel. The selector is left-shifted by 1 so as to skip over the Cb and Cr bytes. Specifically:

```
pixelAddr = (verticalAddr * textureRowLongs + horizontalAddr/2) * 4
+ textureBase
byteSelect= (horizontalAddr & 0x1) << 1</pre>
```

6.13.3.2.3 VQ Addressing VQ address generation occurs in two stages:

- 1. The TextureMap addresses are calculated, and vectors are fetched for 16 pixels.
- The vectors are used to fetch Codebook entries.

6.13.3.2.4 VQ/4:4:4 Addressing The vector fetch phase of VQ/4:4:4 addressing involves first calculating the address of the 32-bit word containing the desired vector, and then after fetching the 32-bit word, selecting the vector from the word by means of a partial barrel shifter. Since the vectors are either nybbles or bytes in 4-bit or 8-bit mode, respectively, there are either 8 or 4 pixels in each 32-bit word. Thus, we need to divide the horizontal address by 8 or 4, respectively, to calculate the 32-bit word address. We then save the LSBs to control the partial barrel shifter and select the desired vector from the fetched 32-bit word. Specifically:

```
(For 4-bit vectors)

vectorAddr = (verticalAddr * textureRowLongs + horizontalAddr/8) *
4 + textureBase

nybbleSelect = horizontalAddr & 0x7

(For 8-bit vectors)

vectorAddr = (verticalAddr * textureRowLongs + horizontalAddr/4) *
4 + textureBase

byteSelect = horiontalAddr & 0x3
```

The Codebook fetch phase of VQ1/4:4:4 addressing is simply a matter of adding the fetched vector address to the codebookBase. Or:

```
pixelAddr = vector + codebookBase
```

6.13.3.2.5 VQ/4:2:2 Addressing VQ/4:2:2 addressing is quite similar to VQ/4:4:4 addressing, but since each vector refers to 2 pixels, the LSB of the horizontal address is used to select the first or second Y from the YCbYCr Codebook entry. Thus, in the vector address calculation, the horizontal address is divided by an additional factor of 2.

Also, a second pixel in the vertical dimension is introduced. Thus, we must use the LSB of the vertical address to select which pixel we are using. Note that the 4:2:2 Codebook is now twice as large as the 4:4:4 Codebook, and Codebook addressing must scaled by a factor of two, accordingly.

So, for the vector fetch phase:

(For 4-bit vectors)

vectorAddr = (verticalAddr/2 * textureRowLongs + horizontalAddr/16)

* 4 + textureBase

nybbleSelect = horiontalAddr/2 & 0x7

(For 8-bit vectors)

vectorAddr = (verticalAddr/2 * textureRowLongs + horizontalAddr/8)

* 4 + textureBase

byteSelect = horiontalAddr/2 & 0x3

The Codebook fetch phase of VQ/4:2:2 addressing now involves making the LSB of the vertical address the LSB of the Codebook fetch address:

pixelAddr = vector * 2 + (verticalAddr & 0x1) + codebookBase

Once the YCbYCr pixel is fetched, it is necessary to select the correct Y byte. This is accomplished by using the LSB of the horizontal address and left-shifting it by 1 so as to skip over the Cb and Cr bytes. Specifically:

byteSelect = (horizontalAddr & 0x1) << 1

6.13.3.3 Address Generation Summary

See Table 6-17 for a listing of all of the address generation formulas.

Table 6-17Address Generation Formula

Data	Size	Address[25:2] (all are word addresses)
yMap entry	32 bits	yMapBase + yMapCount
CelRecord	2 - 12 words	celsBase + celBlockID + celCount
Direct 4:4:4	32 bits	textureBase + ((v * textureRowLongs) + u)
Direct 4:2:2	32 bits 24 used	textureBase + ((v * textureRowLongs) + u/2) {select the correct Y based on u[0]}
VQ8 4:4:4	8 bits 32 bits	vector = textureBase + ((v * textureRowLongs) + u/4) {select the correct byte based on u[1:0]} codeBook = vector + codeBookBase

Table 6-17Address Generation Formula (Continued)

Data	Size	Address[25:2] (all are word addresses)
VQ8 4:2:2	8 bits 32 bits 24 used	vector = textureBase + ((v/2 * texture- RowLongs) + u/8) {select the correct byte based on u[1:0]} codeBook = vector*2 + v[0] + codeBookBase {select the correct Y based on u[0]}
VQ4 4:4:4	4 bits 32 bits	vector: = textureBase + ((v * textureRowLongs) + u/8) {select the correct nibble based on u[2:0]} codeBook = vector + codeBookBase

Note: Where u and v are 8-bit signed numbers.

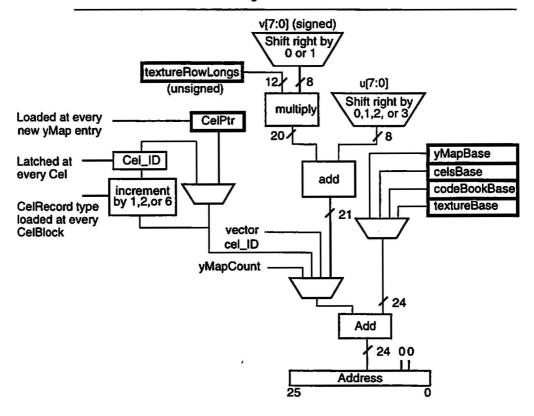


Figure 6-26 Address Generation Block Diagram

6.13.4 Mapping Subunit

The Mapping Subunit takes x and y values and outputs u and v values. The mathematics behind SOLO1's texture space-to-screen space mapping is a linear perspective mapping function. The scanline algorithm used by SOLO1 is as follows:

```
for (y = topLine + topOffset; y < bottomLine - bottomOffset;
V++)
// setup for each new line of Cel
yOffset = yCount - (topLine + topOffset);
u = uStart + yOffset * duRowAdjust;
v = vStart + yOffset * dvRowAdjust;
xLeft = xLeftStart + yOffset * dxLeft;
   xRight = xRightStart + yOffset * dxRight;
// step through x for each scanline
   x = trunc(xLeft);
   while(x < xRight) {
      screen (x,yCount) = TextureMap(u,v);
      u += dux; v += dvx;
      X++;
   }
}
```

6.13.4.1 Mapping Stage 1

The first stage of the Mapping Function takes some of the CelRecord parameters as they are read in and performs the math necessary to get the left edge, right edge, and initial (u,v) values for the current scan line. This has two main advantages:

- It makes it unnecessary to provide separate registers for the CelRecord parameters such as uStart and duRowAdjust, since all we are interested in for this scanline is the uLeftEdge value that is computed from uStart and duRowAdjust.
- 2. It allows us to use the same multiply-accumulate block for these five operations as well as using it for other calculations that occur after the loading of the CelRecord parameters.

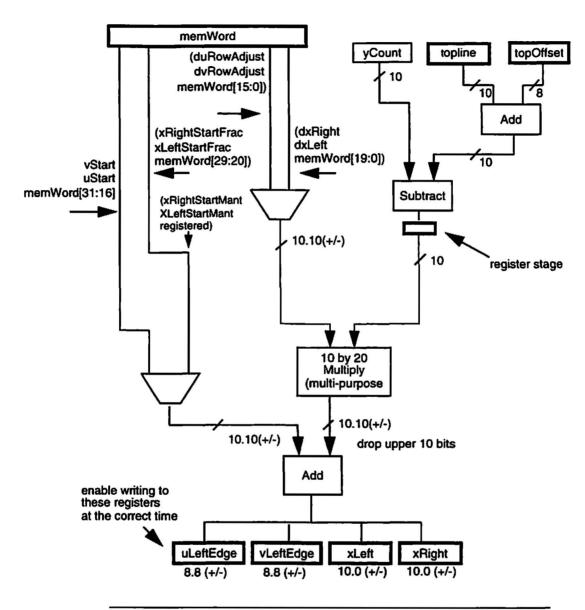


Figure 6-27 Mapping Function Stage 1

6.13.4.2 Stage 2

The second stage of the Mapping Function increments x from xLeftEdge to xRightEdge and computes the u and v values for each x.

Start at x = xLeftEdge While x < xRightEdge:

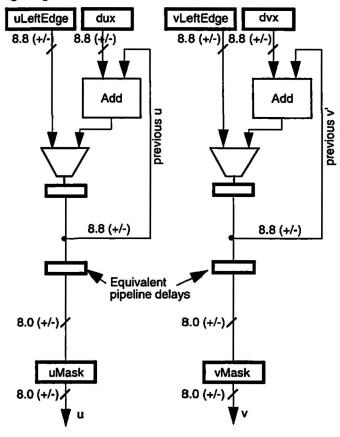


Figure 6-28 Mapping Function Stage 2

6.13.5 Vector Quantitization Subunit

The Vector Quantitization Subunit (fvqUnit) takes the texture space variables (u, v) as inputs and outputs the correct pixel color associated with a given pair of u/v coordinates and the given mode. The pixel color is then sent to the Composite subUnit, where it is written into the scanline FIFO. If the texture is a *Direct Texture*, then the VQ function is essentially bypassed.

6.13.5.1 Vector/Codebook Cache

A 16-entry buffer is included on-chip (see Figure 6-29). This allows the vectors, as well as the Codebook entries, to be fetched from memory in bursts. Each entry contains the four or eight bits for the vector, along with the least significant u and v bits for that vector. The u[0] and v[0] bits are used to help select the correct pixel out of the Codebook for VQ8 4:2:2 mode (Figure 6-30).

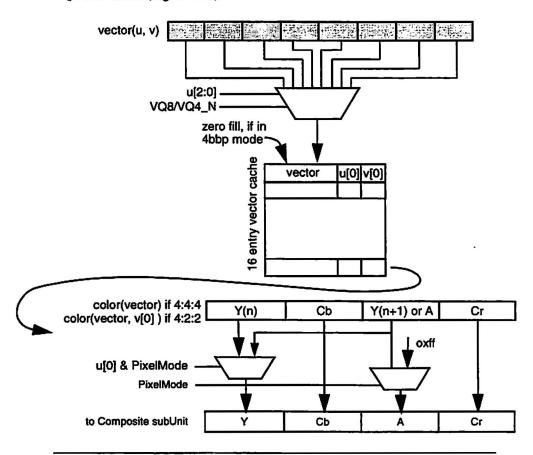


Figure 6-29 Vector Buffer

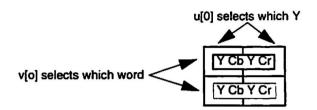


Figure 6-30 Two-by-two Pixel Block in VQ8 4:2:2 Codebook

6.13.5.2 Codebook

The location of the Codebook in SGRAM is pointed to by the codeBookBase parameter in the CelRecord.

For faster performance in VQ4 mode, a 16-entry Codebook can be loaded into the on-chip Codebook RAM. The on-chip Codebook is always used by a microCelRecord, but can also be used by a miniCelRecord or a full CelRecord. A codeBookBase pointer of 0 forces the use of the on-chip codebook. The on-chip Codebook is loaded when a special microCelRecord is issued.

6.13.6 Blend Subunit

The Blend Subunit (foldUnit) is responsible for taking the YCrCb data and blending it into the Scanline Buffer using Alpha.

6.13.6.1 Blending Algorithm

The blend occurs based on the values of Global Alpha, Per-pixel Alpha, and a test of the Y value (see Figure 6-31).

The value of Global Alpha (Aglobal) is set in the celRecord on a per-Cel basis.

The per-pixel Alpha (Aforegnd) is available when using 4:4:4 pixels but not when using 4:2:2 pixel format. For 4:2:2 pixel format, the default value for Aforegnd is 0xff. Also, if the value of the Yforegnd is 0xFF, then the foreground pixel is forced to transparent with an Alpha value of 0.

For the Compositing operation, the pixel already in the Scanline buffer is considered to be the background pixel. The new pixel being blended is considered to be the foreground pixel. The value of Alpha varies from 0 (transparent), which writes 100% of the background pixel back into the scanline buffer, to 1 (opaque), which writes 100% of the foreground pixel into the scanline buffer.

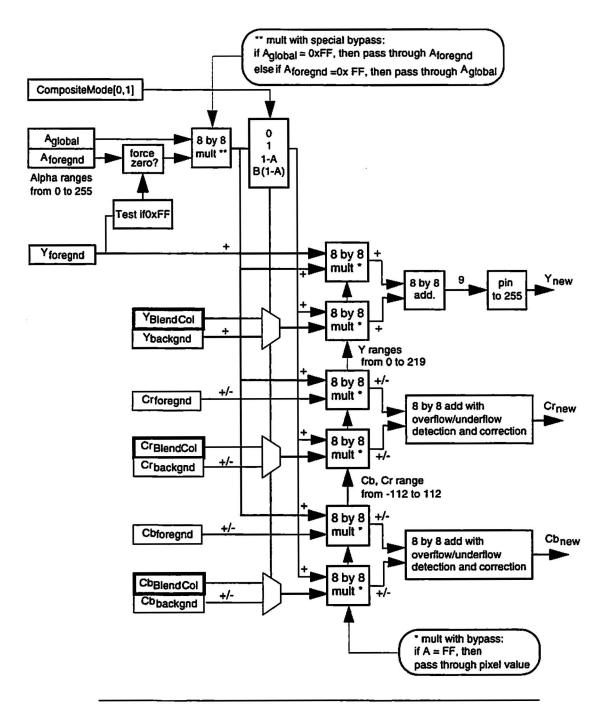


Figure 6-31 Alpha Blend Block Diagram

6.13.6.2 Details of the Multipliers and Adders

The Alpha multipliers have special bypass modes in order to handle the case where A = 0xFF. This is because if we were to multiply 0xFF by a pixel value, that value would be slightly diminished, instead of being the 100% value that was intended. So, when A = 0xFF, the multipliers are bypassed and the full pixel value is passed through.

The adder for the Y channel adds two positive numbers. The overflow condition is detected, and the output forced to 0xFF.

The adder for the chrominance channels adds two signed numbers. The overflow and underflow conditions must be detected. If we are adding a[7:0] and b[7:0] to get the result r[7:0], then the pseudocode to check for overflow would look like:

```
r = a + b;

if (a[7] == 1 and b[7] == 1 and r[7] == 0)

then r = 0x80;

if (a[7] == 0 and b[7] == 0 and r[7] = 1)

then r = 0x7F;
```

This is because overflow occurs only when a and b have the same sign and the sign of the result differs from that of the operands.

6.13.6.3 Alpha Blend Modes

The Alpha Blend operation has four modes that are set in the celMode byte of the CelRecord (see Table 6-18). These modes control the background blending value. The foreground value is always blended with A, where A = Aforegnd * Aglobal.

Table	6-18	Alpha	Blend	Modes
-------	------	-------	-------	-------

Mode	Foregnd Blend	Backgnd Blend	Resulting Pixel
Force Background Transparent	A	0	A* (Foregnd) A* (ForegndColor)
Force Background Opaque	A	1	A * (Foregnd) + (Backgnd) A * (ForegndColor) + (BackgndColor)
Alpha Blend	Α	1-A	A * (Foregnd) + (1-A) * (Backgnd) A * (ForegndColor) + (1-A) * (BackgndColor)
Brighten	A	(1-A) * (BlendColor)	A * (Foregnd) + 2(1-A) A * (Foregnd-Color) + (1-A) * (BlendColor)

Note: (1-A) can be obtained by inverting all the bits of A.

6.13.7 RAM Line Buffer Subunit

There are two internal Scanline buffers in the RAM Line Buffer Subunit (frlbUnit). Note that there is a separate Y, Cb, and Cr value for each pixel in the buffer. Despite the fact that Cb and Cr may be subsampled relative to Y in RAM, they are at full resolution in the Scanline buffer.

Each Scanline buffer is separated into two banks (see Figure 6-32). Since the compositing operation is a read-modify-write operation, this organization allows one pixel to be written every cycle. While one pixel is being read from Bank 0, another pixel can be written to Bank 1 and visa versa (see Figure 6-33).

The organization of the two banks is as follows:

```
Bank 0: Pixel 0, Pixel 1, Pixel 4, Pixel 5, Pixel 8,...
Bank 1: Pixel 2, Pixel 3, Pixel 6, Pixel 7, Pixel 10,...
```

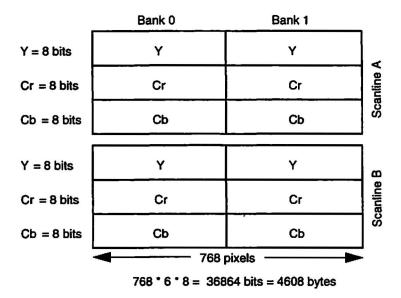


Figure 6-32 Scanline Buffer Organization

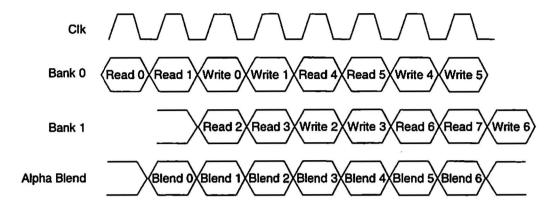


Figure 6-33 One Pixel Composited Every Cycle

The Scanline buffers are composed of asynchronous single-ported memories with a synchronous wrapper added to facilitate chip testing. The synchronous wrapper also helps to simplify the chip timing analysis.

The Scanline buffers form the boundary between the front end of the chip (clocked by SysClk), and the back end of the chip (clocked by VidClk). The Compositing subUnit must write and read the memory, however, the Video Output subUnit only needs to read data from the memory.

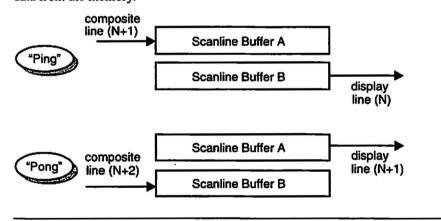


Figure 6-34 Ping-Pong Operation

6.13.8 Video Output Subunit

The Video Output Subunit (fvoUnit) reads the data from the line buffers and presents it to the output pins in either NTSC or PAL format. This subUnit operates using the PIX2XCLK clock.

The 4:4:4 YCrCb data, from the buffers, must be converted to 4:2:2 YCrCb data for output. This is done by averaging two pixels worth of Cr and Cb (see Figure 6-35).

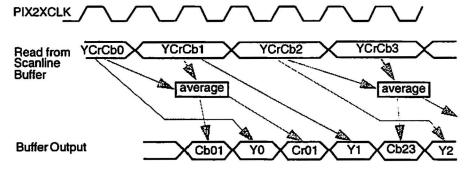


Figure 6-35 Pixel Output

The Video Output subUnit must also shift the values of Y, Cr, and Cb to the ranges appropriate for the dveUnit. In the Scanline buffer, the legal range for Y is [0:219] and the legal range for Cr and Cb is [-112:112] (the reason for this is that representing zero as 0 really helps the math). The Video Encode chip expects Y to be in the range [16:235] and Cr and Cb to be in the range [16:240] where 128 represents zero.

In order to get Y, Cb, and Cr into the correct range, first a floor and/or ceiling function is applied to bring all pixel values back into range. Then the Y value must be shifted by 16 to bring it into the [16:235] range. Similarly, the Cr and Cb values must be shifted by 128 to bring them to the [16:240] range (see Figure 6-36).

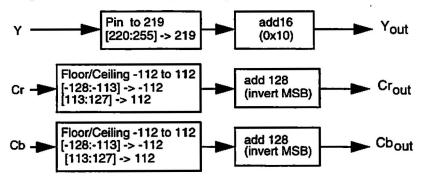


Figure 6-36 Shifting Y, Cb, Cr Values

6.13.8.1 Determining Odd or Even Fields

The Video Output subUnit also uses the HSYNC_N and VSYNC_N signals to create an internal Vertical Counter (VCount) and to determine whether the current field is odd or even.

If HSYNC_N has a falling edge at the same time as VSYNC_N, then the next field is the ODDFIELD, or FIELD 1, otherwise, the next field is EVENFIELD or FIELD 2...

6.13.8.2 The VCount Counter

The VCount counter is reset at the falling edge of VSYNC_N. At every falling edge of HSYNC_N, the counter advances by two. The least significant bit of the counter is determined by the *ODDFIELD* signal. If the field is odd, then the LSB of the counter is 1, otherwise the LSB is 0.

The screen space has the coordinates (0,0) in the middle of the screen, so that the video display is centered for both NTSC and PAL, using the same data structures. However, this means that the counter must reset to a negative number. For NTSC, the vCount resets to (-274). For PAL, the vCount resets to (-332). Figure 6-37 illustrates the relationship between NTSC and vCount numbering. Figure 6-38 illustrates the relationship between PAL and vCount numbering.

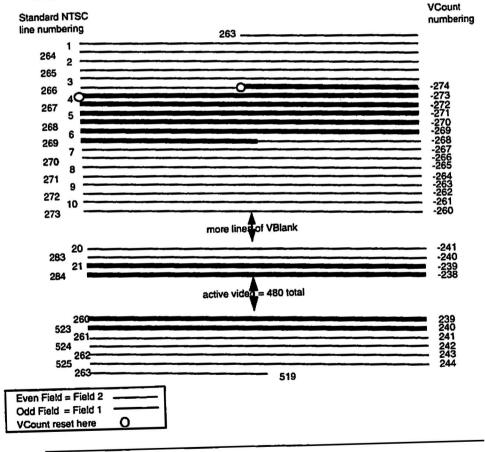


Figure 6-37 NTSC Line Numbering

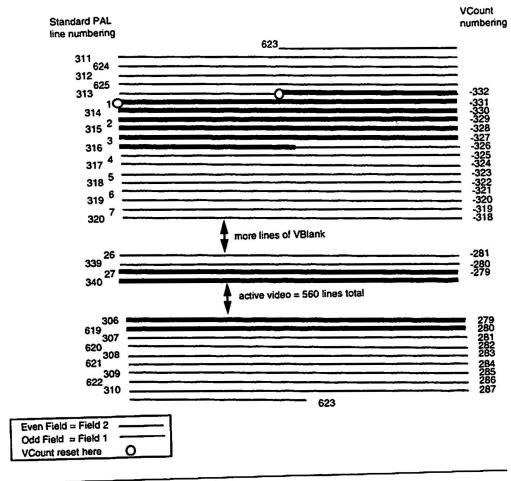


Figure 6-38 PAL Line Numbering

6.13.8.3 TV Display Characteristics

There are several important characteristics worth noting:

1. NTSC and PAL systems overscan. This means that the active video area extends beyond the border of the TV screen, resulting in an Overscan Region that is generally not visible to the viewer and an Underscan Region that is. There is no exact specification for overscan, and it varies from TV to TV. The numbers shown in Figure 6-39 are based on Steve Perlman's experience with NTSC TVs and what he has read about PAL TVs. To be certain that graphics you are displaying are visible to the customer, you should inset it from the TV screen border defined below by about 20 pixels. If you want to guarantee that graphics overscan, you should make sure the graphics extend 20 pixels from the TV screen border. This inset region of the screen is

- often called "safe title" since it is "safely" within the underscan area, in case you wish to display an important title to the viewer.
- Because the overscan is not visible, it is not necessary to composite image data into
 the overscan regions. SOLO1 displays a (specified) solid color during overscan
 intervals in case the customer has a ProScan TV, which can view the overscan video.
- TV video is interlaced. All of the odd lines are scanned out, then all of the even lines
 are scanned out. Each of these raster scans is called a field, and two fields are called a
 frame. NTSC scans at 59.94 fields/sec and 29.97 frames/sec. PAL scans at 50 fields/
 sec and 25 frames/sec.

Figure 6-39 shows the relevant dimensions and timing of the NTSC and PAL systems.

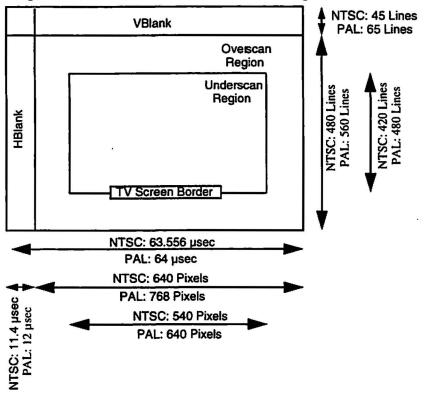


Figure 6-39 NTSC/PAL System Timing

6.14 Miscellaneous/Clock Unit

The mckUnit contains three main submodules:

- mckTest (contains all of the test control logic)
- mckResetAll (contains all of the logic that controls the reset of both SOLO1's internal logic and the external system)
- mckResetCpu (generates the reset sequence required by both the IDT R4640 and the QED RM5230 CPU)

6.14.1 mckTest

The mckTest logic decodes the various test control inputs (TEST_MODE, TEST_SCANEN, GPIO[6:0], CPU_MODECLK and RIO_DINT[1]) and generates the signals listed in Table 6-19 to select the various test modes. The test modes are described in "Test Modes" section, later in this chapter.

Table 6-19 mckTest Test Modes and Enable Signals

Enable Signal	Description
test_scanMode	Enables scan mode. Since scan is not implemented in SOLO1, this signal is unused.
test_triOutMode	Used to tristate all the outputs and bidirectional I/ Os during the TRIOUT testmode, as well as dur- ing the parametric test mode, scan shift mode, and 5V-tolerant disable mode.
test_triOutMode_MISC_LED_0	Special tristate control for the MISC_LED[0] output because it is only tristated during the TRI-OUT testmode, scan shift mode and 5V-tolerant disable mode.
test_triOutMode_PP_DATA	Special tristate control for the PP_DATA bus because it is only tristated during the TRIOUT testmode, parametric test mode, and 5V-tolerant disable mode.
test_5VstbyMode_n	Disables the 5V-tolerant circuitry in the 5V-tolerant I/Os. This circuit draws DC power and must be disabled during <i>Idds</i> testing.
test_paramTestMode	Enables the parametric test chain output to appear on the MISC_LED[0] pin.
test_ramTestMode	Enables the test logic around the embedded RAMs to make the RAMs accessible to the pins.
test_ramTestRamSel[3:0]	Selects which RAM array is to be tested.
test_ramEnCycle[3:0]	Controlled by RIO_DINT[1] and, in turn, controls the RAM enable signal.

Table 6-19 mckTest Test Modes and Enable Signals (Continued)

Enable Signal	Description
test_ramSelPortA	Selects port A or port B of the dual-ported RAMs. It is connected to GPIO[4].
test_pllTestMode	Puts the PLL into test mode.
test_pllClkOutSel	Enables the PLL clock output onto the TEST_SCANEN pin. This feature was removed in SOLO1 Rev. 1.3.
test_pllClkOut	The clock output that is connected to the TEST_SCANEN pin. This feature was removed in SOLO1 Rev. 1.3.
test_dacTestMode[2:0]	Puts each of the DACs into test mode.

Various signals are used during test to guarantee determinstic behavior across the clock boundaries, as well as to shorten the amount of time required to actually enter normal operation. These signals are test_syncInitClocks, test_syncInitRstSm, test_syncShortRstCounters and test_syncShortCounters.

The mckTest unit also drives the chip ID register value onto the mck_BUS_CHIPID bus and on to the busUnit, guaranteeing that these signals are available at the top level for easy FIB or ECO modification.

6.14.2 mckResetAll

This submodule's behavior is described in the reset description section.

6.14.3 mckResetCpu

The IDT R4640 and the QED RM5230 CPUs require an elaborate reset sequence. The various reset states are shown in Table 6-20.

Table 6-20 CPU Reset States

Reset State	Description
BOOTCONFIG	The CPU is out of raw reset and is receiving configuration information serially on the MODEIN input.
ZERO	Configuration information has been transferred; the MODEIN data is being padded with zeros. During the special test modes, this state is shortened to minimize time during test and during simulation.
COLDRESET	One extra cycle between ZERO and RESET states.

Table 6-20 CPU Reset States (Continued)

Reset State	Description		
RESET	CRESET to the CPU is deasserted at the beginning of this state.		
IDLE	SRESET to the CPU is deasserted at the beginning of this state. The CPU is now in normal operating mode.		

See the R4640 or the RM5230 User's Manuals for more information.

6.14.4 Signal Definitions

The mckUnit signals are described in Table 6-21.

Table 6-21 mckUnit Signals

Signal Name	Direction	Description
SYS_2XCLKIN	I	System clock
SYS_PWROK	I	System power OK from power supply
SYS_RSWTCH_N	I	System reset switch input
SYS_RESET_N	0	System reset input
TEST_MODE[1:0]	I	Various test modes
TEST_SCANEN	NO	Scan test enable
CPU_VCCOK	0	Synchronous power-OK signal to CPU
CPU_CRESET_N	0	CPU cold reset
CPU_SRESET_N	0	CPU hard reset
CPU_MODECLK	I	CPU mode clock
CPU_MODEIN	0	CPU mode data input

6.15 PLL Unit

6.15.1 Overview

The PLL Unit (pllUnit) uses a Toshiba Phase-locked Loop device, embedded in SOLO1, to lock the chip's internal clocks to the external system clock. Synchronizing the clocks removes the traditional clock insertion delay variances from the system timing, allowing the system to run at 83.3MHz. The pllUnit dynamically adjusts SOLO1's internal clocks in response to variations in the system speed that result from factors such as temperature changes, voltage, and ASIC processes.

Figure 6-40 provides a functional block diagram of the pllUnit logic.

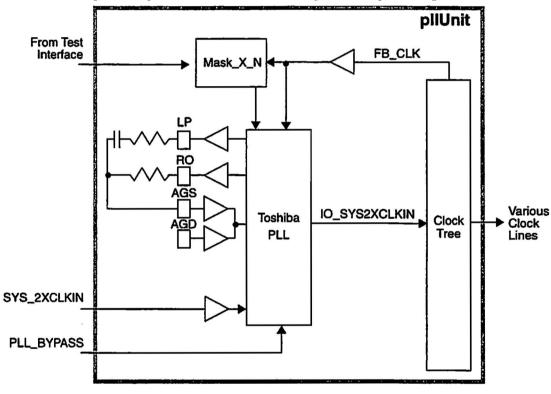


Figure 6-40 pllUnit Functional Block Diagram

6.15.2 Signal Definitions

The pllUnit signals are described in Table 6-1.

Table 6-22pllUnit Signals

Signal Name Direction		Description		
PLL_VAA	I	Analog power		
PLL_BYPASS	I	Bypass PLL circuit- clock goes direct		
PLL_AGS	It	Ground reference		
PLL_AGD	I	Analog ground		
PLL_LP	0	Loop filter out		
PLL_RO	I	Resistor value out		

6.16 Reset Logic

The reset logic is used to generate a synchronized reset signal to the CPU, the other units within the chip, and to the rest of the system components that require a reset signal. The system reset may be triggered by one of four causes: the power supply is powered on or off, the reset switch is closed, a watchdog reset is asserted, or sofware requested a reset. The cause of the reset is preserved across the reset so that software may determine the cause.

In general, all software-visible registers are set to a fixed initial value for any reset. The only exceptions to this are the *Reset Cause* and the *Boot Mode* registers. The Reset Cause register is used to determine the cause of the most recent system reset. This register may be set and cleared by software. Setting any bit of the Reset Cause register immediately triggers a system reset, enabling software to emulate any of the reset causes.

The Boot Mode register is used to reset the CPU with a different boot initialization. Software may request that the default CPU initialization sequence be replaced with one specified in the Boot Mode register after the next reset. This enables software to specify different CPU modes (such as little endian mode).

The reset logic is controlled by a 4-state state machine, shown in Figure 6-41. Upon initial power up, the state machine asynchronously resets to the RESETTING state. When the signal causing the reset is deasserted, the state machine moves to the HOLDRESET state. The state machine stays in this state for 2**25 cycles. The reset logic will assert reset to the rest of the chip and to the system for as long as the state machine is in either the RESETTING or HOLDRESET state. This means that the system reset will be asserted for at least 402ms at 83MHz.

After the HOLDRESET state, the state machine moves to the NORMAL state. This state is the normal operating state of the system. The state machine stays in this state until one of the four reset causes are detected. Once a reset is detected, the state machine moves in to the GONNARESET state. This state indicates that a reset is imminent and cannot be aborted, but the reset signal is not asserted yet. This state can be used to trigger logic used to put the system in a known-good state before the reset occurs. For example, the Smart Card logic will use this state to deactivate any active card before the reset occurs. The GONNARESET state lasts for 2**18 cycles (3.14ms at 83MHz), after which the state machine returns to the RESETTING state. The power supply guarantees that power will remain valid for at least 100ms after the deassertion of SYS_PWROK, ensuring that power will be valid for the entire duration of the GONNARESET state.

Because the logic is required to delay asserting the system reset, rather than asserting the reset immediately following a reset request, special logic is needed to ensure proper initial power up. The logic is shown in the SYS_PWROK circuit in Figure 6-41. This logic must generate an asynchronous reset before SYS_PWROK is initially asserted (when the power supply is initially powered on), but it must not assert an asynchronous reset when SYS_PWROK is dessarted (after power down). An asynchronous reset of the state machine on power-down would force the GONNARESET state to end prematurely.

External circuitry is required to provide a delayed version of the SYS_PWROK signal. A characteristic of this delayed version of SYS_PWROK is that it follows SYS_PWROK for assertion (power-up), but follows the 3.3V supply falling at power-down. The combination of the SYS_PWROK signal, and its delayed version, is used to create the gotInitialPowerOn signal, which only initializes the state machine at initial system power-on.

The initialization of the state machine to the RESETTING state guarantees that the system reset is asserted immediately upon system power on. Without it, there would be the possibility that reset would be asserted 400ms (at 83MHz) after power-on; resulting in bus conflicts during that time.

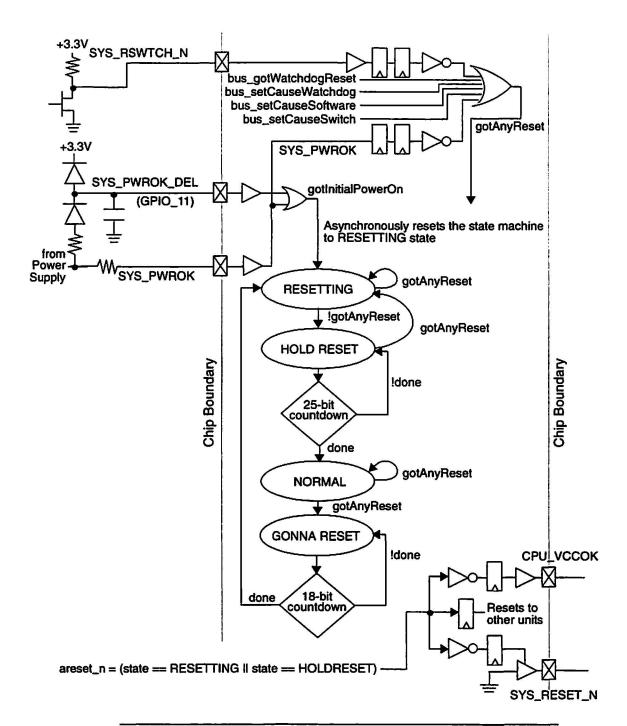


Figure 6-41 Reset Logic

6.17 Test Modes

There are four major test modes, controlled by TEST_MODE[1:0]:

- 00: Normal operation
- 01: Tristate all output drivers
- 10: Test synchronization
- 11: Extended

There are a number of extended test modes, controlled by {CPU_MODECLK,GPIO[3:0]}:

- 0 xxxx scan mode
- 1 0010 5V standby mode
- 1 0011 parametric test mode
- 1 0100 RAM test mode
- 1 0101 PLL test mode
- 1 1000 DAC0 test mode
- 1 1010 DAC1 test mode
- 1 1100 DAC2 test mode
- 1 1110 DACALL test mode

Tristating of output drivers:

When TEST_MODE == 01, all output drivers are tristated. During scan shift, all output drivers are tristated, with the exception of PP_DATA, which is used for scan out. During parametric test, all output drivers are tristated, with the exception of MISC_LED[0], which is the output of the parametric test chain.

6.17.1 Scan Operation

There are two signals that are important during scan:

test_scanMode is active when TEST_MODE == EXTENDED and CPU_MODECLK is low. This mode is used during the entire scan test operation (scan shift, parallel measure and parallel capture).

testScanShiftMode is active only during scan shift. This signal can be used instead of TEST_SCANEN if TEST_SCANEN needs to be qualified with test_scanMode.

6.17.2 Extended Test Modes

There are a number of extended test modes, which are selected by GPIO[3:0] when TEST_MODE = 11:

Scan mode (not implemented in SOLO1)

- · 5V Standby mode
- Parametric Test mode
- · RAM Test mode
- DAC Test modes (4)
- PLL Test mode

Scan mode is for use during scan testing. Scan mode is not just for use during scan shift (when TEST_SCANEN is high), but should also be active during parallel measure and parallel capture.

5V Standby mode is for use during Iddq testing. This mode disables the 5V tolerant circuits, which normally dissipate DC current.

Parametric Test mode is for use to test the Vil/Vih parametric test chain.

Ram Test mode is for use in testing the embedded RAMs. This mode allows direct access of the RAMs from the external pins.

- GPIO[6:5] selects which of the four embedded RAMs is accessed.
- GPIO[4] selects which RAM port is accessed.
- RIO_DINT[0] and RIO_DATA[31:24] drive the RAM address.
- RIO_DINT[2] drives the RAM read/write signal.
- RIO_DATA[23:0] drives the ram data input.
- RIO_WE_N, RIO_ADDR[22:2], and RIO_CE_N[1:0] will have the output of the RAM.

There are four DAC Test modes; 3 for testing each DAC separately and one for test all three DACs simulataneously. These modes allow direct access to the DACs from external pins.

- DIV_DATA is the upper 8 bits of data
- DIV_LLC is the clock.

Currently the lower two bits are not testable.

PLL Test mode is for use in testing the PLL. The special PLL test pins can be accessed from external test pins during this mode.

- VID_DATA[1:0] is the pllMN signal.
- VID_DATA[2] is the pllFS0 signal.
- VID_DATA[3] is the pllFS1 signal.
- VID_DATA[4] is the pllT1 signal.
- VID_DATA[5] is the pllT2 signal.
- VID_DATA[6] is the pllFTS signal.
- RIO_OE_N will be the pllLD signal.

6.18 Electrical Specifications

6.18.1 Signal Descriptions

Table 6-23 SOLO1 Signal Electrical Parameters

Signal	Direction	Buffer Name	Drive (mA)	Comments
AUD_CLK	clkin	IBUF		CMOS level, pullup
MOD_CLK	input	Z1TLCHTHUV12		5V tolerant, pullup
POT_CLK	clkin	IBUF		CMOS level, pullup
AUD_BITCLK	inout	BD4RSTU	4	pullup
MOD_BITCLK	inout	BD4RSTU	4	pullup
AUD_LRCLK	inout	BD4RSTU	4	pullup
MOD_LRCLK	inout	BD4RSTU	4	pullup
CPU_VALOUT_N	input	IBUFU		CMOS level, pullup
CPU_MODECLK	input	IBUF		CMOS level, pullup
TEST_MODE[1:0]	input	IBUFU		CMOS level, pullup
TEST_SCANEN	input	IBUFD		CMOS level, pullup, pulldown
RIO_DINT[3:0]	input	ZITLCHTHUVI2		5V tolerant, pullup
RIO_DEVIORDY	input	ZITLCHTHUVI2		5V tolerant, pullup
RIO_EXPACK_N	input	SMTTU		Schmitt trigger, IV TTL level, pullup
MOD_SDATAIN	input	Z1TLCHTHUVI2		5V tolerant, pullup
AUD_SDATAIN	input	ZITLCHTHUVI2		5V tolerant, pullup
IR_IN	input	ZITLCHTHUVI2		5V tolerant, pullup
PP_SELECT	input	SMTTD		Schmitt trigger, IV TTL level, pulldown
PP_FAULT_N	input	SMTTU		Schmitt trigger, IV TTL level, pullup
PP_ERROR	input	SMTTD		Schmitt trigger, IV TTL level, pulldown
PP_ACK_N	input	SMTTU		Schmitt trigger, IV TTL level, pullup
PP_BUSY	input	SMTTD		Schmitt trigger, IV TTL level, pulldown
SYS_PWROK	input	SMTTD		Schmitt trigger, IV TTL-level, pulldown

Table 6-23 SOLO1 Signal Electrical Parameters (Continued)

Signal	Direction	Buffer Name	Drive (mA)	Comments
SYS_RSWTCH_N	input	SMTTU		Schmitt trigger, IV TTL-level, pullup
CPU_VALIN_N	output	BT16R	16	tri-state, slew rate control
CPU_WRRDY_N	output	BT16R	16	tri-state, slew rate control
CPU_INT_N	output	BT16R	16	tri-state, slew rate control
CPU_VCCOK	output	BT16R	16	tri-state, slew rate control
CPU_CRESET_N	output	BT16R	16	tri-state, slew rate control
CPU_SRESET_N	output	BT16R	16	tri-state, slew rate control
CPU_MODEIN	output	BT16R	16	tri-state, slew rate control
MEM_CKE	output	BT16R	16	tri-state, slew rate control
MEM_CS_N[1:0]	output	BT16R	16	tri-state, slew rate control
MEM_RAS_N	output	BT16R	16	tri-state, slew rate control
MEM_CAS_N	output	BT16R	16	tri-state, slew rate control
MEM_WE_N	output	BT16R	16	tri-state, slew rate control
MEM_ADDR[10:0]	output	BT16R	16	tri-state, slew rate control
MEM_BS	output	BT16R	16	tri-state, slew rate control
MEM_DQM[3:0]	output	BT16R	16	tri-state, slew rate control
RIO_CE_N[1:0]	output	BT16R	16	tri-state, slew rate control
RIO_OE_N	output	BT16R	16	tri-state, slew rate control
RIO_WE_N	output	BT16R	16	tri-state, slew rate control
RIO_DEN_N[3:0]	output	BT16R	16	tri-state, slew rate control
RIO_EXPCLK	output	BT16R	16	tri-state, slew rate control
RIO_EXPEN	output	BT16R	16	tri-state, slew rate control
AUD_SDATA	output	BT16R	16	tri-state, slew rate control
MOD_SDATA	output	BT16R	16	tri-state, slew rate control
PP_SELIN_N	inout	BD16RSTU	16	pullup
PP_AUTOFD_N	inout	BD16RSTU	16	pullup
PP_STROBE_N	inout	BD16RSTU	16	pullup
PP_INIT_N	output	BT16R	16	tri-state, slew rate control
PP_DIR	output	BT16R	16	tri-state, slew rate control
MISC_LED[2:0]	output	BT16R	16	tri-state, slew rate control

Table 6-23 SOLOI Signal Electrical Parameters (Continued)

Signal	Direction	Buffer Name	Drive (mA)	Comments
RIO_ADDR[22:2]	inout	BD16RC	16	CMOS level
CPU_AD[31:0]	inout	BD16C	16	CMOS level
CPU_CMD[6:0]	inout	BD16C	16	CMOS level
GPIO[15:7]	inout	BD16RSTU	16	pullup
GPIO[6:0]	inout	ZIBD8THUVI2	8	5V tolerant, pullup
DIV_DATA[7:0]	inout	BD8RSTU	8	pullup
RIO_DATA[31:16]	inout	BD16RSTU	16	pullup
RIO_DATA[15:0]	inout	Z1BD8THUVI2	8	5V tolerant, pullup
MEM_DATA[31:0]	inout	BD16RCU	16	pullup
IR_OUT	inout	BD16RSTU	16	pullup
IR_CLK	inout	Z1BD8THUVI2	8	5V tolerant, pullup
SYS_RESET_N	output	BT24R	24	tri-state, slew rate control
SMC_INSERT_N	inout	BD16RSTU	16	pullup
SMC_PEN_N	inout	Z1BD8THUVI2	8	5V tolerant, pullup
SMC_RESET_N	inout	Z1BD8THUVI2	8	5V tolerant, pullup
SMC_CLK	inout	Z1BD8THUVI2	8	5V tolerant, pullup
SMC_DATA	inout	Z1BD8THUVI2	8	5V tolerant, pullup
UART_RXD	inout	BD16RSTU	16	pullup
UART_CTS_N	inout	BD16RSTU	16	pullup
UART_DCD_N	inout	BD16RSTU	16	pullup
UART_TXD	inout	BD16RSTU	16	pullup
UART_RTS_N	inout	BD16RSTU	16	pullup
UART_DTR_N	inout	BD16RSTU	16	pullup
PP_DATA[7:0]	inout	BD16RSTU	16	pullup
IIC_DATA	inout	Z1BD8THUVI2	8	5V tolerant, pullup
IIC_CLK	inout	Z1BD8THUVI2	8	5V tolerant, pullup
ID_DATA	inout	BD8RSTU	8	pullup
DIV_HS	inout	BD16RSTU	16	pullup
DIV_VS	inout	BD16RSTU	16	pullup
DIV_LLC	input	SMTTU		Schmitt trigger, IV TTL-level, pullup

Table 6-23 SOLO1 Signal Electrical Parameters (Continued)

Signal	Direction	Buffer Name	Drive (mA)	Comments
DIV_LRCLK	inout	BD16RSTU	16	puliup
DIV_BCLK	inout	BD16RSTU	16	pullup
DIV_SDATA	inout	BD16RSTU	16	pullup
VID_DATA[7:0]	inout	BD16RSTU	16	pullup
VID_HSYNC_N	inout	Z1BD8THUVI2	8	5V tolerant, pullup
VID_VSYNC_N	inout	Z1BD8THUVI2	8	5V tolerant, pullup
SYS_2XCLKIN	input	IBUF		CMOS level, pullup
PLL_BYPASS	input	IBUF		CMOS level, pullup
PLL_RO	output	BDRV	N/A	analog signal
PLL_LP	output	BDRV	N/A	analog signal
PLL_AGD	input	AGND	N/A	analog signal
PLL_AGS	input	AGS	N/A	analog signal
PLL_VAA	input	VAA	N/A	analog signal
DAC_CRCBAOUT	output	BDRV	N/A	analog signal
DAC_YAOUT	output	BDRV	N/A	analog signal
DAC_COMPAOUT	output	BDRV	N/A	analog signal
DAC_VREF	input	AGS	N/A	analog signal
DAC_YVBS	input	AGS	N/A	analog signal
DAC_CRCBVBS	input	AGS	N/A	analog signal
DAC_COMPVBS	input	AGS	N/A	analog signal
DAC_CRCBAPVD	input	DVDD13A	N/A	analog signal
DAC_COMPAPVD	input	DVDD13A	N/A	analog signal
DAC_YAPVD	input	DVDD13A	N/A	analog signal
DAC_CRCBAPVS	input	DVSSTA	N/A	analog signal
DAC_COMPAPVS	input	DVSSTA	N/A	analog signal
DAC_YAPVS	input	DVSSTA	N/A	analog signal

6.18.2 Pin Assignments

Table 6-24 SOLO1 Pin Assignments

Pin No.	Signal Name	Pin No.	Signal Name
T24	AUD_BITCLK	T23	AUD_CLK
T25	AUD_LRCLK	U26	AUD_SDATA
R26	AUD_SDATAIN	A10	CPU_AD[0]
A15	CPU_AD[10]	C14	CPU_AD[11]
B14	CPU_AD[12]	A14	CPU_AD[13]
D10	CPU_AD[14]	C10	CPU_AD[15]
B10	CPU_AD[16]	A9	CPU_AD[17]
A8	CPU_AD[18]	A5	CPU_AD[19]
A17	CPU_AD[1]	A2	CPU_AD[20]
B 3	CPU_AD[21]	A4	CPU_AD[22]
C5	CPU_AD[23]	B5	CPU_AD[24]
D6	CPU_AD[25]	C6	CPU_AD[26]
B6	CPU_AD[27]	B7	CPU_AD[28]
B8	CPU_AD[29]	B17	CPU_AD[2]
C8	CPU_AD[30]	D8	CPU_AD[31]
B18	CPU_AD[3]	B19	CPU_AD[4]
A20	CPU_AD[5]	D17	CPU_AD[6]
D16	CPU_AD[7]	B16	CPU_AD[8]
C15	CPU_AD[9]	A18	CPU_CMD[0]
A19	CPU_CMD[1]	C17	CPU_CMD[2]
C16	CPU_CMD[3]	A16	CPU_CMD[4]
B15	CPU_CMD[5]	D14	CPU_CMD[6]
C7	CPU_CRESET_N	B9	CPU_INT_N
C18	CPU_MODECLK	B4	CPU_MODEIN
A7	CPU_SRESET_N	A6	CPU_VALIN_N
A3	CPU_VALOUT_N	C9	CPU_VCCOK
C19	CPU_WRRDY_N	J26	DAC_COMPAOUT
K26	DAC_COMPAPVD	M23	DAC_COMPAPVS
K25	DAC_COMPVBS	L23	DAC_CRCBAOUT
H25	DAC_CRCBAPVD	J25	DAC_CRCBAPVS

Table 6-24 SOLO1 Pin Assignments (Continued)

Pin No.	Signal Name	Pin No.	Signal Name
K24	DAC_CRCBVBS	H23	DAC_NC[0]
L22	DAC_NC[10]	M22	DAC_NC[11]
L24	DAC_NC[12]	L25	DAC_NC[13]
L26	DAC_NC[14]	E25	DAC_NC[1]
G24	DAC_NC[2]	J22	DAC_NC[3]
F26	DAC_NC[4]	G25	DAC_NC[5]
G26	DAC_NC[6]	K22	DAC_NC[7]
J24	DAC_NC[8]	H26	DAC_NC[9]
K23	DAC_VREF	J23	DAC_YAOUT
E26	DAC_YAPVD	F25	DAC_YAPVS
H24	DAC_YVBS	C20	DIV_BCLK
B24	DIV_DATA[0]	A24	DIV_DATA[1]
C23	DIV_DATA[2]	B23	DIV_DATA[3]
A23	DIV_DATA[4]	C22	DIV_DATA[5]
B22	DIV_DATA[6]	A22	DIV_DATA[7]
B21	DIV_HS	B25	DIV_LLC
B20	DIV_LRCLK	D20	DIV_SDATA
C21	DIV_VS	C24	GPIO[0]
B2	GPIO[10]	C1	GPIO[11]
D24	GPIO[12]	R24	GPIO[13]
AB23	GPIO[14]	AC4	GPIO[15]
D21	GPIO[1]	D22	GPIO[2]
U23	GPIO[3]	W23	GPIO[4]
AA24	GPIO[5]	AC25	GPIO[6]
A21	GPIO[7]	A25	GPIO[8]
BI	GPIO[9]	DI	ID_DATA
A26	IIC_CLK	C26	IIC_DATA
AB3	IR_CLK	AC1	IR_IN
AF26	IR_OUT	R2	MEM_ADDR[0]
RI	MEM_ADDR[10]	Pl	MEM_ADDR[1]
N4	MEM_ADDR[2]	N3	MEM_ADDR[3]

Table 6-24 SOLO1 Pin Assignments (Continued)

Pin No.	Signal Name	Pin No.	Signal Name
P2	MEM_ADDR[4]	NI	MEM_ADDR[5]
N2	MEM_ADDR[6]	M1	MEM_ADDR[7]
M2	MEM_ADDR[8]	Li	MEM_ADDR[9]
R3	MEM_BS	TI	MEM_CAS_N
L2	MEM_CKE	U1	MEM_CS_N[0]
K2	MEM_CS_N[1]	AA3	MEM_DATA[0]
WI	MEM_DATA[10]	Y2	MEM_DATA[11]
Y 1	MEM_DATA[12]	AA2	MEM_DATA[13]
AA1	MEM_DATA[14]	AA4	MEM_DATA[15]
El	MEM_DATA[16]	F2	MEM_DATA[17]
F1	MEM_DATA[18]	G2	MEM_DATA[19]
Y3	MEM_DATA[1]	G1	MEM_DATA[20]
H2	MEM_DATA[21]	H1	MEM_DATA[22]
J2	MEM_DATA[23]	M3	MEM_DATA[24]
L3	MEM_DATA[25]	K4	MEM_DATA[26]
K3	MEM_DATA[27]	J3	MEM_DATA[28]
Н3	MEM_DATA[29]	W3	MEM_DATA[2]
G3	MEM_DATA[30]	F3	MEM_DATA[31]
V3	MEM_DATA[3]	V4	MEM_DATA[4]
U3	MEM_DATA[5]	T4	MEM_DATA[6]
T3	MEM_DATA[7]	V1	MEM_DATA[8]
W2	MEM_DATA[9]	U2	MEM_DQM[0]
V2	MEM_DQM[1]	J1	MEM_DQM[2]
K1	MEM_DQM[3]	T2	MEM_RAS_N
P3	MEM_WE_N	AB1	MISC_LED[0]
AB2	MISC_LED[1]	AC2	MISC_LED[2]
AF25	MOD_BITCLK	AC23	MOD_CLK
AE25	MOD_LRCLK	AF24	MOD_SDATA
AD23	MOD_SDATAIN	E5	NO_CON[0]
AB5	NO_CON[1]	AF2	NO_CON[2]
AE26	NO_CON[3]	A12	PLL_AGD

Table 6-24 SOLO1 Pin Assignments (Continued)

Pin No.	Signal Name	Pin No.	Signal Name
C13	PLL_AGS	D11	PLL_BYPASS
D13	PLL_LP	C11	PLL_NC[0]
C12	PLL_NC[1]	E13	PLL_RO
B12	PLL_VAA	B11	PLL_VDD
A13	PLL_VSS	T26	POT_CLK
AB26	PP_ACK_N	Y25	PP_AUTOFD_N
AB25	PP_BUSY	U25	PP_DATA[0]
U24	PP_DATA[1]	V26	PP_DATA[2]
V25	PP_DATA[3]	V24	PP_DATA[4]
W26	PP_DATA[5]	W25	PP_DATA[6]
W24	PP_DATA[7]	AC26	PP_DIR
AB24	PP_ERROR	Y24	PP_FAULT_N
AA26	PP_INIT_N	AC24	PP_SELECT
AA25	PP_SELIN_N	Y26	PP_STROBE_N
AD3	RIO_ADDR[10]	AD5	RIO_ADDR[11]
AE6	RIO_ADDR[12]	AC11	RIO_ADDR[13]
AC13	RIO_ADDR[14]	AE13	RIO_ADDR[15]
AC14	RIO_ADDR[16]	AE14	RIO_ADDR[17]
AC15	RIO_ADDR[18]	AC3	RIO_ADDR[19]
AE2	RIO_ADDR[20]	AEI	RIO_ADDR[21]
AE3	RIO_ADDR[22]	AF14	RIO_ADDR[2]
AD14	RIO_ADDR[3]	AF13	RIO_ADDR[4]
AD13	RIO_ADDR[5]	AF11	RIO_ADDR[6]
AE10	RIO_ADDR[7]	AF4	RIO_ADDR[8]
AF3	RIO_ADDR[9]	AC20	RIO_CE_N[0]
AC21	RIO_CE_N[1]	AE5	RIO_DATA[0]
AF10	RIO_DATA[10]	AF12	RIO_DATA[11]
AF9	RIO_DATA[12]	AF7	RIO_DATA[13]
AF5	RIO_DATA[14]	AD4	RIO_DATA[15]
AE15	RIO_DATA[16]	AE16	RIO_DATA[17]
AF17	RIO_DATA[18]	AF18	RIO_DATA[19]

Table 6-24 SOLO1 Pin Assignments (Continued)

Pin No.	Signal Name	Pin No.	Signal Name
AE7	RIO_DATA[1]	AD19	RIO_DATA[20]
AC18	RIO_DATA[21]	AF16	RIO_DATA[22]
AF15	RIO_DATA[23]	AC16	RIO_DATA[24]
AD17	RIO_DATA[25]	AD18	RIO_DATA[26]
AE19	RIO_DATA[27]	AE18	RIO_DATA[28]
AE17	RIO_DATA[29]	AE9	RIO_DATA[2]
AD16	RIO_DATA[30]	AD15	RIO_DATA[31]
AE12	RIO_DATA[3]	AE11	RIO_DATA[4]
AF8	RIO_DATA[5]	AF6	RIO_DATA[6]
AE4	RIO_DATA[7]	AD6	RIO_DATA[8]
AE8	RIO_DATA[9]	AF21	RIO_DEN_N[0]
AD2	RIO_DEN_N[1]	ADI	RIO_DEN_N[2]
AF19	RIO_DEN_N[3]	AE21	RIO_DEVIORDY
AD21	RIO_DINT[0]	AF1	RIO_DINT[1]
AF20	RIO_DINT[2]	AE22	RIO_DINT[3]
AC22	RIO_EXPACK_N	AF22	RIO_EXPCLK
AD22	RIO_EXPEN	AE20	RIO_OE_N
AD20	RIO_WE_N	C4	SMC_CLK
C3	SMC_DATA	C2	SMC_INSERT_N
D2	SMC_PEN_N	D3	SMC_RESET_N
D12	SYS_2XCLKIN	Al	SYS_PWROK
B26	SYS_RESET_N	C25	SYS_RSWTCH_N
E24	TEST_MODE[0]	D25	TEST_MODE[1]
D26	TEST_SCANEN	AD25	UART_CTS_N
AE24	UART_DCD_N	AF23	UART_DTR_N
AE23	UART_RTS_N	AD24	UART_RXD
AD26	UART_TXD	R25	VDD3[0]
E12	VDD3[10]	F22	VDD3[11]
AC17	VDD3[1]	U4	VDD3[2]
D7	VDD3[3]	All	VDD3[4]
L4	VDD3[5]	Y23	VDD3[6]

Table 6-24 SOLO1 Pin Assignments (Continued)

Pin No.	Signal Name	Pin No.	Signal Name
D18	VDD3[7]	AC8	VDD3[8]
G4	VDD3[9]	E9	VDD[0]
P22	VDD[10]	W5	VDD[11]
J4	VDD[12]	T5	VDD[13]
AA23	VDD[14]	D5	VDD[15]
E15	VDD[16]	E17	VDD[17]
E22	VDD[18]	AB9	VDD[19]
V22	VDD[1]	E7	VDD[20]
F24	VDD[21]	H22	VDD[22]
AB20	VDD[2]	AC12	VDD[3]
L5	VDD[4]	AB16	VDD[5]
G5	VDD[6]	R4	VDD[7]
M4	VDD[8]	E3	VDD[9]
P26	VID_DATA[0]	P25	VID_DATA[1]
P24	VID_DATA[2]	N26	VID_DATA[3]
P23	VID_DATA[4]	N25	VID_DATA[5]
N24	VID_DATA[6]	N23	VID_DATA[7]
M25	VID_HSYNC_N	M26	VID_VSYNC_N
AB14	VSS1[0]	P5	VSS1[10]
F23	VSS1[11]	E16	VSS1[12]
E23	VSS1[13]	AB13	VSS1[14]
AD12	VSS1[15]	M5	VSS1[16]
ADII	VSS1[17]	E18	VSS1[18]
R22	VSS1[19]	AD8	VSS1[1]
K5	VSS1[20]	AB12	VSS1[21]
E8	VSS1[22]	AC7	VSS1[23]
D23	VSS1[24]	E6	VSS1[25]
N22	VSS1[26]	E14	VSS1[27]
E11	VSS1[28]	AB17	VSS1[29]
AB18	VSS1[2]	D7	VSS1[30]
N5	VSS1[31]	AB7	VSS1[32]

Table 6-24 SOLO1 Pin Assignments (Continued)

Pin No.	Signal Name	Pin No.	Signal Name
AB4	VSS1[33]	Y5	VSS1[34]
E10	VSS1[35]	D4	VSS1[36]
Y4	VSS1[37]	J5	VSS1[38]
AB22	VSS1[39]	AB19	VSS1[3]
AC6	VSS1[40]	U5	VSS1[41]
AB6	VSS1[42]	AD10	VSS1[43]
AC9	VSS1[44]	E20	VSS1[45]
U22	VSS1[46]	F5	VSS1[47]
AB11	VSS1[48]	R5	VSS1[49]
AB21	VSS1[4]	W22	VSS1[50]
Y22	VSS1[51]	AC5	VSS1[52]
AB10	VSS1[53]	E4	VSS1[54]
H5	VSS1[55]	AD9	VSS1[5]
E21	VSS1[6]	AA22	VSS1[7]
AC19	VSS1[8]	AB8	VSS1[9]
V23	VSS2[0]	F4	VSS2[1]
E2	VSS2[2]	M24	VSS2[3]
AA5	VSS2[4]	V5	VSS2[5]
E19	VSS2[6]	G23	VSS2[7]
H4	VSS3[0]	D9	VSS3[10]
G22	VSS3[11]	AC10	VSS3[1]
B13	VSS3[2]	P4	VSS3[3]
T22	VSS3[4]	R23	VSS3[5]
AB15	VSS3[6]	D15	VSS3[7]
D19	VSS3[8]	W4	VSS3[9]

Table 6-25 SOLO1 Signal Descriptions

Signal name	Direction	Description
CPU interface		
CPU_AD[31:0]	ΙΟ	CPU address and data bus
CPU_CMD[6:0]	ΝO	CPU command bus
CPU_VALOUT_N	I	Address, command, and data valid strobe from CPU
CPU_VALIN_N	0	Data and command valid strobe back to CPU
CPU_WRRDY_N	0	CPU write ready signal
CPU_INT_N	0	CPU interrupt line
CPU_VCCOK	0	Power OK signal to CPU (synchronous)
CPU_CRESET_N	0	CPU cold reset
CPU_SRESET_N	0	CPU hard reset
CPU_MODECLK	I	CPU mode clock for sending boot mode data to CPU
CPU_MODEIN	0	Boot mode data to CPU
Phase Locked Loop In	terface	
PLL_VAA	I	PLL analog supply
PLL_BYPASS	I	PLL bypass. When '1', SYS_2XCLKIN bypasses the PLL
PLL_AGS	I	PLL analog power
PLL_AGD	I	PLL analog ground
PLL_LP	0	PLL loop filter out
PLL_RO	0	PLL loop resistor value out
Misc and Clock Interfa	ice	
SYS_2XCLKIN	I	System clock
SYS_PWROK	I	System power OK from power supply
SYS_RSWTCH_N	I	System reset switch input
SYS_RESET_N	0	System reset input
GPIO[11]	0	System warm reset. When low, SOLO1 assumes a cold power-on and reset will apply immediately.
TEST_MODE[1:0]	I	Selects test mode (see "Test Mode" section).

Table 6-25 SOLO1 Signal Descriptions (Continued)

Signal name	Direction	Description
TEST_SCANEN	I/O	Enable scan testing. Also can be used to look at a divide-by-2 version of the internal SYS_2XCLKIN for PLL debug.
Memory Interface		
MEM_CKE	0	SDRAM clock enable
MEM_CS_N[1:0]	0	SDRAM chip selects
MEM_RAS_N	0	SDRAM row address strobe
MEM_CAS_N	0	SDRAM column address strobe
MEM_WE_N	0	SDRAM write enable
MEM_ADDR[10:0]	0	SDRAM address
MEM_BS	0	SDRAM bank select
MEM_DQM[3:0]	0	SDRAM byte enables
MEM_DATA[31:0]	1/O	SDRAM data bus
RIO (ROM) Interface		
RIO_CE_N[1:0]	0	ROM bank selects
RIO_OE_N	0	ROM and RIO device output enable
RIO_WE_N	0	ROM and RIO device write enable
RIO_ADDR[22:2]	ΛO	ROM and RIO device address
RIO_DATA[31:0]	I/O	ROM and RIO device data
RIO_DEN_N[3:0]	0	RIO expansion device enables
RIO_DINT[3:0]	I	RIO expansion interrupts
RIO_DEVIORDY	I	RIO expansion IO ready signal
RIO_EXPCLK	0	RIO expansion device clock
RIO_EXPEN	0	WebTV Port Expansion Bus enable
RIO_EXPACK_N	I	WebTV Port Expansion Bus ack
Video Interface		
POT_CLK	I	Video display pixel clock. 4X pixel rate when on-chip video encoder is used. 2X pixel rate when off-chip video encoder is used.
VID_HSYNC_N	ΙΟ	Horizontal sync; input from off-chip encoder or output from on-chip encoder.
VID_VSYNC_N	ИО	Vertical sync input; input from off-chip encoder or output from on-chip encoder.

Table 6-25 SOLO1 Signal Descriptions (Continued)

Signal name	Direction	Description
VID_DATA[7:0]	ΝO	Video data bus to off-chip encoder
Digital Video Encoder	Interface	
DAC_YVBS	I	VBS for the Y DAC
DAC_CRCBVBS	I	VBS for the CrCb DAC
DAC_COMPVBS	Ī	VBS for the Comp DAC
DAC_YAOUT	0	AOUT for the Y DAC
DAC_CRCBAOUT	0	AOUT for the CrCb DAC
DAC_COMPAOUT	0	AOUT for the Comp DAC
DAC_YAPVD	0	APVD for the Y DAC
DAC_CRCBAPVD	0	APVD for the CrCb DAC
DAC_COMPAPVD	0	APVD for the Comp DAC
DAC_YAPVS	0	APVS for the YDAC
DAC_CRCBAPVS	0	APVS for the CrCb DAC
DAC_COMPAPVS	0	APVS for the Comp DAC
DAC_VREF	I	DAC voltage reference
Digital Video Input Into	erface	
DIV_DATA[7:0]	ΙΛΟ	DIVIT data input
DIV_HS	ΝO	DIVIT control signals
DIV_VS	I/O	DIVIT control signals
DIV_LLC	I	DIVIT control signals
DIV_LRCLK	I/O	DIVIT audio LRCLK
DIV_BCLK	ΙΛΟ	DIVIT audio bit clock
DIV_SDATA	I/O	DIVIT audio serial data
Audio Interface		
AUD_CLK	1	Audio Clock input (11.28Mhz)
AUD_SDATAIN	I	Audio serial data input (ADC)
AUD_BITCLK	0	Audio bit clock for serial data
AUD_LRCLK	0	Left/right clock
AUD_SDATA	0	Audio serial data output (DAC)
Soft Modem Interface		
MOD_CLK	I	Modem master clock

Table 6-25 SOLO1 Signal Descriptions (Continued)

Signal name	Direction	Description
MOD_SDATAIN	I	Modem serial data input (ADC)
MOD_BITCLK	0	Modern bit clock for serial data
MOD_LRCLK	0	Modem frame sync (like LRCLK)
MOD_SDATA	0	Modem serial data output (DAC)
Smartcard and UART	Interface	
SMC_DATA	I/O	SmartCard Data (inout)/Uart RXD
SMC_INSERT_N	I/O	SmartCard Inserted (input)/Uart CTS
SMC_RESET_N	ΙΟ	SmartCard Reset (output)/Uart RTS
SMC_PEN_N	NO	SmartCard Power Enable (output)/Uart DTR
SMC_CLK	NO	SmartCard Clock (output)/Uart TXD
UART_RXD	NO	Uart receive data (input)/DATA/GPIO
UART_CTS_N	I/O	Uart clear to send (input)/INSERT/GPIO
UART_DCD_N	NO	Uart data carrier detect (input)/GPIO
UART_TXD	I/O	Uart transmit data (output)/CLK/GPIO
UART_RTS_N	I/O	Uart request to send (output)RESET/GPIO
UART_DTR_N	NO	Uart data terminal ready (output)PEN/GPIO
I/O Devices Interface		
D_DATA	NO	ID chip data line
IR_IN	I	Infrared receiver data
IR_CLK	NO	Infrared receiver clock
IR_OUT	NO	Infrared data output
PP_DATA[7:0]	NO	Parallel Port Data
PP_ACK_N	I	Parallel Port Ack
PP_BUSY	I	Parallel Port Busy
PP_ERROR	I	Parallel Port Error
PP_SELECT	ī	Parallel Port Select
PP_FAULT_N	I	Parallel Port Fault
PP_SELIN_N	I/O	Parallel Port SelectIn
PP_AUTOFD_N	NO	Parallel Port Auto Feed
PP_STROBE_N	NO	Parallel Port Strobe
PP_INIT_N	0	Parallel Port init

Table 6-25 SOLO1 Signal Descriptions (Continued)

Signal name	Direction	Description
PP_DIR	0	Parallel Port direction
MISC_LED[2:0]	0	LED outputs
GPIO[15:0	I/O	General-purpose I/O
IIC_CLK	ΙΛΟ	IIC clock
IIC_DATA	I/O	IIC data line

6.19 Mechanical Specifications

The SOLO1 packaging dimensions are shown in the following two figures. Figure 6-42 illustrates the early Type 1 packaging, used until early 1998. Figure 6-43 shows the current, Type 2 packaging.

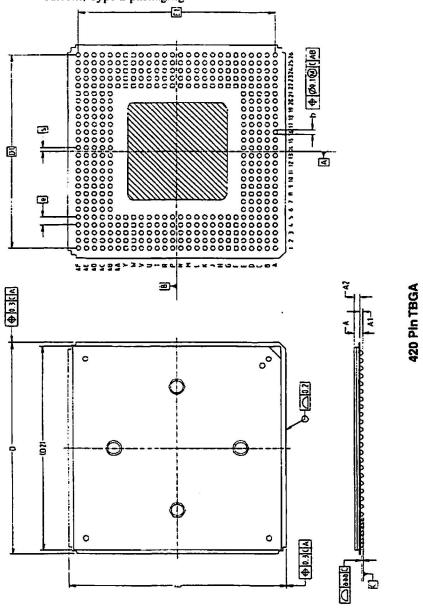


Figure 6-42 Type 1 Packaging Dimensions

Table 6-26 Type 1 Packaging Dimensions

Package Type (Package Code)	420 Pin TBGA (BGA420-T-3535-1.27A)						
Symbol	Millimeters						
	Min	Nom	Max				
A	1.2	1.4	1.6				
Al	0.5	0.6	0.7				
A2	0.7	0.8	0.9				
b	0.60	0.75	0.90				
D	34.8	35.0	35.2				
D1	8	31.75 BSC					
D2	ć	33.6 REF					
E	34.8	35.0	35.2				
El		31.75 BSC	92				
е		1.27 BSC					
s		0.635 BSC					
aaa	PR 100	0.15					

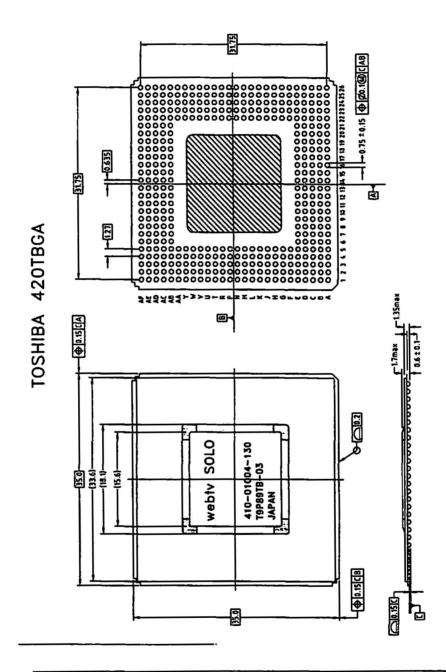


Figure 6-43 Type 2 Packaging Dimensions

The following foldout shows both top and bottom views of the SOLO1 pinout.

	***		************		***********	************			***********		***********		10000000000000000000000000000000000000	THOUT (top v	sessessessesses ior, balls not	**************** Visible on t	644444444444 him mide): 8644444444444	************		 	************	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,				44984688888	**********
26			SYS_RESET_H		TEST_SCANES	DAC_YAPVD	NO_CON	No_con		DAC_COMPAGUT S					VID DATA[0]		POT_CLX	AUD_SDATA			H_\$60872_99		PP_ACK_N	PP_DIR	UART_TXD	NO_CON	IR_COT
25		GP10[8]	DIV_LLC	SYS_REWICH_H	TEST_HOUR[1]	NO_CON	DAC_YAPVS	310_COH D	AC_CROBAPVD D	AC_CRCBAFVS	DAC_CONTVIS			VID_DATA(5)	VID_DATA[1]	ADDI	AUD_LECLE	PP_DATA(0)	PP_DATA(3)	PP_DATA[6]	PP_AUTOFD_H	PP_SEZ_TH_H	17,005Y	GP10[6]	UART_CTS_H	MOD_LECLE	
24	, 0	DIV_DATA(1)	DIV_DATA[0]	GB10[0]	GP10[12]	[0]200K_T22T	VDD	30_C08	DAC_YVBS	NO_COR	DAC_CRCSVSS	MO_COM	VSS2	AID DYLY(e)	VID_DATA(2)	GPIO(13)	AUD_BITCLE	PP_DATA(1)	PP_DATA(4)	PP_DATA[7]	PP_FAULT_H	CPIO(S)	PP_ERROR	PP_SELECT	WART_RED	UART_DCD_H	HOD_SDATA
23	: 0	OIV_DATA[4]	DIV_DATA[3]	DIV_DATA(2)	ARRÍ	V2S1	VSS1	VSS2	360_COH	DAC_YAOUT	DAC_VREP D	AC_CROSMOUT S	_COGRAPVS	VID_DATA[7]	VID_DATA(4)	(22V	AUD_CLE	CPIO[3]	VS52	GP10[4]	VD03	VDD	GP10[14]	MOD_CLX	HIATAGR_DOM	WART_RTS_N	CART_DTS_N
22	: 0	DIV_DATA(7)	DIV_DATA(6)	DIV_DATA(\$)	GP10(2)	VDD	ADD3	VSS3	VDO	HO_COM	NO_CON	NO_CON	NO_CON	VES1	VDO	VSS1	VSSI	VSS1	VDO	VSS1	V\$51	VSS1	122V	LTO_EXPACK_H	RIO_EXPEN	RIO_DINT(3)	RIO_EXPCLE
21	1	GP10(7)	DIV_XS	DIV_VS	CP10(1)	V221									41											RIO_DEVICEDY	
20		CPO_AD(S)	DIV_LECUE	DIV_BCLE	DIV_SDATA	VEST																		RIO_CE_H(0)	RIO_WZ_N		RIO_DINT[2]
19		CPU_CAD(1)		CSO MOCECTY	VSS3	A223																	VSS1			RIO_DATA [27]	
17		CPU_CRD(V)	CMC*D(3)	CFG_CIØ(2)	CPG_AD[6]	VOD.																	VSS1 W			RIO_DATA[28]	
16		CPU CHO(4)	CPU_AD(8)	CPU_CDC(3)	CPU_AD[7]	Vas1																				RIO_DATA[17]	0.000
15		CPU_AD[10]	CPU_CMD(5)	(1) (2,0(2)	VSSJ	VDO																				RIO_DATA(16)	
16		CPU_AD[13]	CPU_AD[12]	CPQ_AD[11]	C50 C00[6]	V\$\$1																				RIO_ADDR(17)	
13		PLL_VES	VSSS	PLL_AGS	FLLLEP	PLL_RO																	VSS1 R	110_ADDR(14)	RIO_ADDR(5)	RIO_ADDR(15)	RIO_ADDR[4]
12		PLL_JGD	PLL_VAA	HO_COM	SYS_ZXCLKIN	VD03																	VSS1	VDD	VSS1	RIO_DATA(3)	RIO_DATA(11)
11	1	VDC3	PLL_VD0	NO_CCH	PLL_DYPASS	V\$51																	VSS1 8	110_ADOR(13)	VSS1	RIO_DATA(4)	RIO_ADDR (6)
10	:	CPU_AD(0)	CPU_AD(16)	CPU_AD[15]	CPU_AD[14]	VSS1																	VSS1	CZSV	VSS1	RIO_ADCR(7)	RIO_DATA(10)
,		CPU_AD(17)	CMO_INT_H	CBO_ACCOX	VSS3	VDD																	VDD	VS\$1	AZZ3	RIO_DATA[2]	RIO_DATA[12]
		CPU_AD[18]			CPU_AD(31)	V251																	VSS1	VD03	VSS1	RIO_DATA(9)	RIO_DATA(5)
7		U_SRESET_N		CPU_CRESET_H	VD03	VDO																	VSS1	VSS1		RIO_DATA[1]	
•		30_ANTIN'N			-	VSS1													-				VSS1			RIO_ADOR(12)	
5		CPG_AD(19)	CPU_AD[24]	CPG_AD(23)	VED	150_COR	VSS1	V20	VSS1	VS31	V951	V00	VS\$1	vssi	V551	VSS1	VDD	VSS1	V\$52	VDO	VSS1	V852	NO_COR			RIO_DATA(0)	
•		GEG_AD(22) GEGALGGT_H		DEC.CLX	VSUL	V2\$1	₩32	ADC3	C22V	ADD 1	CDC_CDATA(26)	VD03	ADD	KENUADOR(2)	CZSV	VDD	HODELDATA(6)	V003	KEN_DATA(4)	E22V	A221	KEX_DATA(15)	V351	GP101121 8	10_DATA[15]	RIO_DATA[7]	RIO_ADDR[8]
2		= =	CLO TO [Set]		-	WAS W		-	THE PART NO.		- mail 171 v	- n- 1361 .	*						WENT DATE: (2)	www.co.es.(3)	MEN PARE (11						
1		CDG 101201	CD101101		SE MIN N		ELDATA(D1) M						-	HODE ADDRESS	HENCHELIN	200 000	HER DATA (7)		HEN_DATA(3)			HEN_DATA(0)	IR_CLE R	110_ADOR(19) 8		RIO_ADDR[22]	
	1	CPU_AD(20) 5YS_MGKK	100000	SHC_DISTRAT_D GP10(11)	וויומראצ	ASS3 N	ERLDATA(31) M ERLDATA(17) M ERLDATA(18) M	DATA (19) N	DLDATA[21] :	SEEL_DATA(23)	100 (23 JH(33)	NO.COL	K(NDDE(8)	MDC/ADDR(6)	162CV508(4)	1623(JADOR [0]	HODY_RAS_R	MESC DÓR(0)	1031_DQH[1]	HEN_DATA(9)	KEN_DATA(11)	HEN_DATA(0)	IR_CLE R	110_ADCR(19) # HISC_LED(2) #	10_DEC.H(1)	R10_ADDR(20)	HO_CON
	1		100000	SHC_DISTRIC_D	וויומראצ	ASS3 N	22LDATA[17] 10	DATA (19) N	DLDATA[21] :	SEEL_DATA(23)	100 (23 JH(33)		K(NDDE(8)			1623(JADOR [0]	HODY_RAS_R	MESC DOM(0)		HEN_DATA(9)	KEN_DATA(11)	HEN_DATA(0)	IR_CLE R	110_ADCR(19) # HISC_LED(2) #	10_DEC.H(1)		HO_CON
	•		100000	GP10(11)	ID_DATA	ASS3 N	22LDATA[17] 10	DATA (19) N	DLDATA[21] :	929(_DASA(23) 929(_DQH(2)	KENTOOH(3)	NO.COL	K(NDDE(8)	MDC/ADDR(6)	162CV508(4)	1623(JADOR [0]	HERUPAS JE	MESC DÓR(0)	1031_DQH[1]	HEN_DATA(9)	KEN_DATA(11)	MEN_DATA [0] MEN_DATA [13] MEN_DATA [14]	IR_CLE R MISC_LED(1) MISC_LED(0)	110_ADCR(19) # HISC_LED(2) #	10_DEC.H(1)	R10_ADDR(20)	HO_CON
•		SYS_MOCK	@10[9]	GPIO(11)	D DATA	VSS2 16 HBH_DATA [16] 10 E	224_DATA[17] 16 284_DATA[18] 16 P	ERLEATA(19) N ERLEATA(20) N G	EDL_DATA[21] E	1031_0ATD.(23) 1031_0QH(2) J	KERF**(3)	HERLETE HERLETE L	HULDOR(8) HULDOR(7)	HEDCADOR(6) HEDCADOR(5) H	162(_100R[4] 162(_100R[1] (MECULDOR(0) MECULDOR(10) R	HEN_RAS_H HEN_CAS_H T	A A A A A A A A A A A A A A A A A A A	MEN_DON(1) MEN_DATA(8)	HEH_DATA(9)	KERL DATA [11]	MEN_DATA (0) MEN_DATA (13) MEN_DATA (14) AA	IR_CLE R HISC_LED(0) HISC_LED(0)	HISC_LED(19) H HISC_LED(2) H IR_IM H AC	77 20 DEF N(S) 30 DEF N(S)	RIO_ADDR(20) RIO_ADDR(21) AE	HO_CON RIO_DINT{1}
	0000	SYS_MOOK A	B G310(9)	C C C C C C C C C C C C C C C C C C C	D DATA	VSS2 10 1001_DATA(16) 10 E	234_DATA(17) NO 234_DATA(18) NO P	EPL_DATA [19] N ERL_DATA [20] N G	EDI_DATA[21] : EDI_DATA[22] E	929(_DASSA(23) 3439(_DQH(2) 3	1001_C2_H(1) H331_DQH(3) X	100-CX2	H LLDOR(7)	NEDCADOR(6) NEDCADOR(5) N	MENTALISMENT OF THE PARTY OF TH	MER(ADOR(0) OD(ADOR(10) R	HERLENSJI T T seessessesses	0 0 0 1005(_DOH(0)	MENLESHELLS	MENI_DATA(9) SEL_DATA(10) W	HERLENATA [11] HERLENATA [12] Y	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA	IR_CLK R HISC_LED(1) AB	HISC_LED(2) H HISC_LED(2) H IR_IM H AC	70_DEM_H[2]	RIO_ADDR[20] RIO_ADDR[21] AE	NO_CON RIO_DINT(1) AF
	0000	SYS_MOOK A	B G310(9)	C C C C C C C C C C C C C C C C C C C	D DATA	VSS2 16 HBH_DATA [16] 10 E	234_DATA(17) NO 234_DATA(18) NO P	EPL_DATA [19] N ERL_DATA [20] N G	EDI_DATA[21] : EDI_DATA[22] E	929(_DASSA(23) 3439(_DQH(2) 3	1001_C2_H(1) H331_DQH(3) X	100-CX2	H LLDOR(7)	NEDCADOR(6) NEDCADOR(5) N	MENTALISMENT OF THE PARTY OF TH	MER(ADOR(0) OD(ADOR(10) R	HERLENSJI T T seessessesses	0 0 0 1005(_DOH(0)	MENLESHELLS	MENI_DATA(9) SEL_DATA(10) W	HERLENATA [11] HERLENATA [12] Y	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA	IR_CLK R HISC_LED(1) AB	HISC_LED(2) H HISC_LED(2) H IR_IM H AC	70_DEM_H[2]	RIO_ADDR[20] RIO_ADDR[21] AE	NO_CON RIO_DINT(1) AF
	0000	SYS_MOOK A	@ZZG[9]	C C C C C C C C C C C C C C C C C C C	ENC. MENUTA ID. DATA D	VSS2 16 IGDC_DATA(146) 10 E	234_DATA(17) NO 234_DATA(18) NO P	EPL_DATA [19] N ERL_DATA [20] N G	EDI_DATA[21] : EDI_DATA[22] E	929(_DASSA(23) 3439(_DQH(2) 3	KEN_DOH(3) X	100-CX2	H LLDOR(7)	NEDCADOR(6) NEDCADOR(5) N	MENTALISMENT OF THE PARTY OF TH	MER(ADOR(0) OD(ADOR(10) R	HERLENSJI T T seessessesses	0 0 0 1005(_DOH(0)	MENLESHELLS	MENLEATA(9)	HERLENTA [11] HERLENTA [12] Y	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA	IR_CLK R HISC_LED(1) AB	HISC_LED(2) H HISC_LED(2) H IR_IM H AC	.co.per.u(2) .co.per.u(2)	RIO_ADDR[20] RIO_ADDR[21] AE	NO_CON RIO_DINT(1) AF
	0000	572_NGGK A 1411441441	@310[8]	CPIO(11)	D DATA (VSS2 16 1650_DATA(16) 15 E 164499999999999	201_DATA[17] NO 201_DATA[18] NO P	PH_DATA(19) N DH_DATA(20) N G 090000000000000000000000000000000000	EDC_DATA[21] : ECC_DATA[22] E	224_DATA(23) 229_DQM(2) 3	KEN_DOH(3) X	IOD_CEE	H_LECOR[8] H_LECOR[7] H SOLO P	NECLADOR(6) NOTE THE PROPERTY OF THE PROPERTY	MON_ADDR[4] MON_ADDR[1] I P Insertal passes m view, balls views settlesses	MEN_ADDR[0] R Reseaseseseseseseseseseseseseseseseseses	TE SIGN):	MEH_DOM(6)	MENUDON(1) MENUDON(1) V	MENLEATA(9)	HERLENTA [11] HERLENTA [12] Y	MEM_DATA(0) MEM_DATA(13) MEM_DATA(14) AA	IR_CLE R MISC_LED(1) MISC_LED(0) AB	120_ADDR(19)	.co.per.u(2) .co.per.u(2)	RIO_ADDR[20] RIO_ADDR[21] AE	NO_CON RIO_DIRT(1) AP
	: 5	STE_MRCK A ICOCCUMANT ICCCUMANT ICCCUMAN	@310[8]	CPIO(11) C ORRESPONDO CPIO(11) C DIV_DATA(1) DIV_DATA(0)	D DATA (VSS2 MSM_DATA(16) MS E 00000000000000000000000000000000000	224_DATA[17] M 234_DATA[18] M 7 10414441444444444444444444444444444444	CPU_AD(S) DIV_LECTA DIV_LECTA DIV_LECTA DIV_LECTA DIV_LECTA	EDI_DATA[22] E ********************************	### DATE (23) #### DATE (23) ####################################	IGBL_DGH[3] IGBL_DGH[3] X (4999999999999999999999999999999999	CSG_CDG[4]	H_LEDER[8] H SELECTION SEL	NDC_ADDR(6) NDC_ADDR(5) N NSSESSESSESSESSESSESSESSESSESSESSESSESSE	NEX_ACOR[4] P Interpolations of the control of th	REPLACE (0) R Resistant and a state of the	HERLENS JE T ALEXANDER STREET ALEXANDER STREET	CPU_DDK[0]	MENL DOM(1) MENL DATA(8) V CPU_AD(17)	EST_DATA(9) ESTELDATA(10) ESTELDATA(10) ESTELDATA(10) ESTELDATA(10) ESTELDATA(10) ESTELDATA(10) ESTELDATA(10) ESTELDATA(10)	HEN_DATA[11] Y OBSIGNATA[12] Y CRESTORNATA CPU_SRESET_A	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA *********************************	IR_CLE R HISC_LED(1) AB	EC_ADDR(19) R REC_LED(2) R RC_HH R AC	AD COUNTY HE STORY H	RIO_ADDR[20] RIO_ADDR[21] AE	NO_CON RIO_DINT(1) AP
A	: 5	STE_MORCE A IIIIIIIII IIII_CLE TS_RESET_H IIII_DATA	©377770 ©310(8)	CPIO(11) C ORRESPONDO CPIO(11) C DIV_DATA(1) DIV_DATA(0)	ID_DATA D DATA D DATA D DATA D DATA D D D DATA D D D DATA D D D D D D D D D D D D D D D D D D D	VSS2 MSM_DATA(16) MS E 00000000000000000000000000000000000	224_DATA(17) MC P 10414419444444444444444444444444444444	CPU_AD(S) DIV_LECTA DIV_LECTA DIV_LECTA DIV_LECTA DIV_LECTA	EDC_DATA[21] E EDC_DATA[22] E EDC_DATA[22] CPC_QDD[1] CPC_DD[4]	J J J J J J J J J J J J J J J J J J J	EDU_DQH[3] KDU_DQH[3] K CHU_DQH[3] CHU_DQH[3]	100_CXE L CPU_CRD(4) CPU_CRD(4)	H_LDDR[8] H seessees SOLO P: 10_LD[10] (G_LD[5]	NDC_ADDR(6) NDC_ADDR(5) N NSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS	NEX_ACOR[4] P IGDE_GOOR[1] II IGDE_GOO	REPLACE PLACE PLAC	HERE, CAS, II	10 (15) (16) (16) (16) (16) (16) (16) (16) (16	MENL DOM(1) MENL DATA(8) V CPU_AD(17) CPU_JMT_J	EST_DATA(9) ESTELDATA(10) ESTELDATA(10) ESTELDATA(10) ESTELDATA(10) ESTELDATA(10) ESTELDATA(10) ESTELDATA(10) ESTELDATA(10)	HEN DATA [11] Y COLUMN TO THE	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA *********************************	IR_CLE R MISC_LED(1) MISC_LED(0) AB **********************************	IIO,ADDR(19) R NISC_LED(2) R IR_IN R AC CPU_AD(22) C CPU_MODEIM	AD COUNTY HE STORY H	RIO_ADDR[20] RIO_ADDR[21] AE **********************************	NO_CON RIO_DIRT(1) AP SYS_PARGE GPIO(9)
A	: S:	STE_MORCE A IIIIIIIII IIII_CLE TS_RESET_H IIII_DATA	GPIO(9) D GPIO(8) GPIO(8) DIV_LLC STR_RENTCE(N TEST_MOGE(1)	CPIO(11) C DIV_DATA(1) DIV_DATA(0) CPIO(0)	ID_DATA D DIV_DATA(4) DIV_DATA(2) DIV_DATA(2)	VSS2 MODE_DATA[16] MODE_DATA[16] MODE_DATA[7] DIV_DATA[6] DIV_DATA[5]	224_DATA[17] NO P P P P P P P P P P P P P P P P P P P	CPU_AD(S) DIV_ECTA DIV_ECTA DIV_ECTA DIV_ECTA	EDC_DATA[21] E EDC_DATA[22] E EDC_DATA[22] CPC_QDD[1] CPC_DD[4]	J J J J J J J J J J J J J J J J J J J	EDU_DQH[3] KDU_DQH[3] K CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3]	100_CXE L 100_ADCS[9] CPU_CRD[4] CPU_CRD[4] CPU_DRD[8]	H_LDDR(F) H SULP P: SULP P: GLDD(10) GLDD(5) SULP P: GLDD(5) SULP P: GLDD(5)	NDC_ADDR(6) NDC_ADDR(5) N NSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS	NEX_ACOR[4] P IGDE_GOOR[1] II IGDE_GOO	REPLACE PLACE PLAC	HERE CAS II T SECOND	100 (0) (0) (0) (0) (0) (0) (0) (0) (0) (MENL DOM(1) MENL DATA(8) ***********************************	EDIL DATA (9) E E E E E E E E E E E E E	HEM. DATA [11] Y *********************************	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA MANAGEMENT CONTROL C	IR_CLE R HISC_LED(1) HISC_LED(0) AB **********************************	IIO,ADDR(19) R NISC_LED(2) R IR_IN R AC CPU_AD(22) C CPU_MODEIM	AD COUNTY HE STORY H	RIO_ADDR(21) AE **********************************	NO_CON RIO_DIRT(1) AP STS_PARGE GPIO(9) GPIO(11)
A	: S:	A IIICCIA	GPIO(9) D SESSION SE	CPIO(11) C OPENSATE P OPENSATE OF OPENSATE OPENSATE OF OPENSATE OPENSA	DIV_DATA(4) DIV_DATA(4) DIV_DATA(2) VSS1 VSS1	VSS2 MCDC_DATA(16) MCDC_DATA(16) MCDC_DATA(7) DIV_DATA(6) DIV_DATA(5) GPIO(2) VDD	224_DATA[17] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[17] NO P 224_DATA[18]	CPI_AD(5) DIV_BCATA DIV_BCATA DIV_BCATA	EDI_DATA[21] E EDI_DATA[22] E CPU_CDD[1] CPU_DD[4] CPU_DD[4] CPU_DD[4] VSS3	J J J J J J J J J J J J J J J J J J J	EDU_DQH[3] KDU_DQH[3] K CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3]	100_CXE 100_ADCS[9] 1 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100 100 100 100 100 100 100 100 10	H_LDDR(F) H SULP P: SULP P: GLDD(10) GLDD(5) SULP P: GLDD(5) SULP P: GLDD(5)	NDC_ADDR(6) NDC_ADDR(5) N NDC_ADDR(5) N NDC_ADDR(5) CPG_AD(13) CPG_AD(12) CPG_AD(12) CPG_AD(12)	MEX_ACOR[4] ACCC_ACOR[1] ACCC_A	REMINISTRATE OF THE PROPERTY O	MEDICAS JI T SECRETARISMENT SECRETARISMENT VEDS PLI_VED PLI_JYPASS	CPU_AD[14]	MENL DOM(1) MENL DATA(8) ***********************************	EDIL DATA (9) E E E E E E E E E E E E E	NEM. DATA [11] Y *********************************	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA CPU_VALIN_N CPU_AD(27) CPU_AD(26) CPU_AD(26)	IR_CLE R MISC_LED(1) MISC_LED(0) AB **********************************	IIO_ADDR(19) R IR_IM R AC CPU_AD[22] C CPU_AD[22] C CPU_MODELM VSS1 VSS2	AD	RIO_ADDR(21) AE **********************************	RIO_DINT[1] AP STS_PARCK GPIO[9] GPIO[11] ID_DATA EXEL_DATA[16] MESL_DATA[18]
A	: S	A IIICCIA	GPIO(9) D STELLANTCH, M TEST_MODE(1) DAC_YAPVE MO_COM	CPIO(11) C CPIO(11) C CPIO(11) CPIO(10) CPIO(12) CPIO(1	DIV_DATA(2) VSS1 VSS2 VSS2	VSS2 MCDC_DATA(16) MCDC_DATA(16) MCDC_DATA(7) DIV_DATA(6) DIV_DATA(5) GPIO(2) VDD VDD3 VSS3	224_DATA[17] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[17] NO P 224_DATA[18]	CPI_AD(5) DIV_BCATA DIV_BCATA DIV_BCATA	EDI_DATA[21] E EDI_DATA[22] E CPU_CDD[1] CPU_DD[4] CPU_DD[4] CPU_DD[4] VSS3	J J J J J J J J J J J J J J J J J J J	EDU_DQH[3] KDU_DQH[3] K CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3]	100_CXE 100_ADCS[9] 1 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100 100 100 100 100 100 100 100 10	H_LDDR(F) H SULP P: SULP P: GLDD(10) GLDD(5) SULP P: GLDD(5) SULP P: GLDD(5)	NDC_ADDR(6) NDC_ADDR(5) N NDC_ADDR(5) N NDC_ADDR(5) CPG_AD(13) CPG_AD(12) CPG_AD(12) CPG_AD(12)	MEX_ACOR[4] ACCC_ACOR[1] ACCC_A	REMINISTRATE OF THE PROPERTY O	MEDICAS JI T SECRETARISMENT SECRETARISMENT VEDS PLI_VED PLI_JYPASS	CPU_AD[14]	MENL DOM(1) MENL DATA(8) ***********************************	EDIL DATA (9) E E E E E E E E E E E E E	NEM. DATA [11] Y *********************************	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA CPU_VALIN_N CPU_AD(27) CPU_AD(26) CPU_AD(26)	IR_CLE R MISC_LED(1) MISC_LED(0) AB **********************************	IIO,ADDR(19) R IR_IM R AC CPU_AD[22] CPU_MODEIN SMC_CLX VSS1 VSS2 VDO3 8	AD	RIO_ADOR(20) RIO_ADOR(21) AE **********************************	RIO_DINT[1] AP STS_MARCK GPIO[9] GPIO[11] EXEL_DATA[16] MESL_DATA[20]
A B C D E F G H	: S	STE_MOCK A IIIC_CLK YE_RESET_M IIC_DATA IIC_DATA EXC_YAP/O MO_COM MO_COM MO_COM	GPIO(9) DIV_LLC SYS_RAPTCH_M TEST_MODE(1) MO_COM DAC_YAPVS MO_COM DAC_COMPAND	CPIO(11) C CPIO(11) C CPIO(11) CPIO(10) CPIO(10) CPIO(12) CPIO(1	DIV_DATA(2) VSS1 VSS2 HO_COM	VSS3 MCDC_DATA(16) MCDC_DATA(16) MCDC_DATA(16) MCDC_DATA(5) MCDC_DATA(224_DATA[17] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[17] NO P 224_DATA[18]	CPI_AD(5) DIV_BCATA DIV_BCATA DIV_BCATA	EDI_DATA[21] E EDI_DATA[22] E COLUMN (1) COLUMN (1)	J J J J J J J J J J J J J J J J J J J	EDU_DQH[3] KDU_DQH[3] K CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3]	100_CXE 100_ADCS[9] 1 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100 100 100 100 100 100 100 100 10	H_LDDR(F) H SULP P: SULP P: GLDD(10) GLDD(5) SULP P: GLDD(5) SULP P: GLDD(5)	NDC_ADDR(6) NDC_ADDR(5) N NDC_ADDR(5) N NDC_ADDR(5) CPG_AD(13) CPG_AD(12) CPG_AD(12) CPG_AD(12)	MEX_ACOR[4] ACCC_ACOR[1] ACCC_A	REMINISTRATE OF THE PROPERTY O	MEDICAS JI T SECRETARISMENT SECRETARISMENT VEDS PLI_VED PLI_JYPASS	CPU_AD[14]	MENL DOM(1) MENL DATA(8) ***********************************	EDIL DATA (9) E E E E E E E E E E E E E	NEM. DATA [11] Y *********************************	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA CPU_VALIN_N CPU_AD(27) CPU_AD(26) CPU_AD(26)	IR_CLE R MISC_LED(1) MISC_LED(0) AB *********************************	CPU_MODEIN SHE_CLX CPU_MODEIN VSS1 VSS2 VSS2 VSS3 VSS3	AD	RIO_ADOR(20) RIO_ADOR(21) AE *********************************	RIO_DINT(1) AP SESSESSESSESSESSESSESSESSESSESSESSESSE
A B C D E P G H J	: S: T: : T: : : : : : : : : : : : : : :	STE_MOOK A SISSESSED AS SITC_DATA SITC_DATA SITC_DATA SITC_COSS SITC_C	GPIO(9) D DIVILIC STG_RSWTCH_N TEST_MODE(1) MO_CON DAC_TAPVS DAC_CREBAPVS DAC_CREBAPVS	GPIO(11) C SERVENCE SERVENCE	DIV_DATA(4) DIV_DATA(4) DIV_DATA(2) VSS1 VSS1 VSSS SS_COS	VSS3 MCDC_DATA(16) MCDC_DATA(16) MCDC_DATA(16) MCDC_DATA(17) MCDC_DATA(1	224_DATA[17] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[17] NO P 224_DATA[18]	CPI_AD(5) DIV_BCATA DIV_BCATA DIV_BCATA	EDI_DATA[21] E EDI_DATA[22] E COLUMN (1) COLUMN (1)	J J J J J J J J J J J J J J J J J J J	EDU_DQH[3] KDU_DQH[3] K CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3]	100_CXE 100_ADCS[9] 1 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100 100 100 100 100 100 100 100 10	H_LDDR(F) H SULPTION SULPTION (U_LD(10)	NDC_ADDR(6) NDC_ADDR(5) N NDC_ADDR(5) N NDC_ADDR(5) CPG_AD(13) CPG_AD(12) CPG_AD(12) CPG_AD(12)	MEX_ACOR[4] ACCC_ACOR[1] ACCC_A	REMINISTRATE OF THE PROPERTY O	MEDICAS JI T SECRETARISMENT SECRETARISMENT VEDS PLI_VED PLI_JYPASS	CPU_AD[14]	MENL DOM(1) MENL DATA(8) ***********************************	EDIL DATA (9) E E E E E E E E E E E E E	NEM. DATA [11] Y *********************************	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA CPU_VALIN_N CPU_AD(27) CPU_AD(26) CPU_AD(26)	IR_CLE R MISC_LED(1) MISC_LED(0) AB *********************************	CPU_MODEIN VSS1 VSS2 VSS2 VSS2 VSS2 VSS3 VSS3 VSS3 VSS3 VSS3	AD	RIO_ADOR(20) RIO_ADOR(21) AE *********************************	RIO_DINT(1) AP STS_PAROK GPIO(9) GPIO(1) HED_DATA(16) MED_DATA(20) MED_DATA(22) MED_DATA(22) MED_DATA(22)
A B C D E P G H J	: S: T: : T: : : : : : : : : : : : : : :	STE_MGGK A STE_MGGK STE_MGGK STE_CLX STE_CL	GPIO(9) D STOLENION STOLENION TEST_MODE(1) MO_COM DAC_CREAPIN DAC_CREAPIN DAC_CREAPINE DAC_CREAPINE	GPIO(11) C SHIPPERSON GPIO(11) DIV_DATA(1) GPIO(12) TEST_MOGE(0) VDO MO_COM DAC_TVES HO_COM DAC_CRESVES	DIV_DATA(4) DIV_DATA(4) DIV_DATA(2) VSS1 VSS1 VSS2 DO_COM DAC_VKEP	VSS2 MEDICATA (16) MEDICATA (16) MEDICATA (16) MEDICATA (17) MEDICATA (1	224_DATA[17] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[17] NO P 224_DATA[18]	CPI_AD(5) DIV_BCATA DIV_BCATA DIV_BCATA	EDI_DATA[21] E EDI_DATA[22] E COLUMN (1) COLUMN (1)	J J J J J J J J J J J J J J J J J J J	EDU_DQH[3] KDU_DQH[3] K CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3]	100_CXE 100_ADCS[9] 1 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100_CXE[9] 100 100 100 100 100 100 100 100 100 10	H_LDDR(F) H SULPTION SULPTION (U_LD(10)	NDC_ADDR(6) NDC_ADDR(5) N NDC_ADDR(5) N NDC_ADDR(5) CPG_AD(13) CPG_AD(12) CPG_AD(12) CPG_AD(12)	MEX_ACOR[4] ACCC_ACOR[1] ACCC_A	REMINISTRATE OF THE PROPERTY O	MEDICAS JI T SECRETARISMENT SECRETARISMENT VEDS PLI_VED PLI_JYPASS	CPU_AD[14]	MENL DOM(1) MENL DATA(8) ***********************************	EDIL DATA (9) E E E E E E E E E E E E E	NEM. DATA [11] Y *********************************	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA CPU_VALIN_N CPU_AD(27) CPU_AD(26) CPU_AD(26)	IR_CLE R MISC_LED(1) MISC_LED(0) AB *********************************	EC_DATA(26)	AD	RIO_ADOR[20] RIO_ADOR[21] AE *********************************	RIO_DIRT(1) AP STS_PAROK GPIO(9) GPIO(11) IO_DATA MEN_DATA(16) MEN_DATA(20) MEN_DATA(22) MEN_DOM(2)
A B C D E P G H J R L	: ST : TI : DAV	STE_MGGK A STE_MGGK STE_MGGK STE_CLX STE_CL	GPIO(9) D OPIO(9)	GPIO(11) C SELECTION OF THE SELECTION	DIV_DATA(4) DIV_DATA(4) DIV_DATA(2) VSS1 VSS1 VSS2 DAC_YAOUT DAC_VEEP DAC_CRCBAOUT	VSS2 MEDICATA (16) ME E ********************************	224_DATA[17] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[17] NO P 224_DATA[18]	CPI_AD(5) DIV_SCATA DIV_SCATA DIV_SCATA	EDI_DATA[21] E EDI_DATA[22] E COLUMN (1) COLUMN (1)	J J J J J J J J J J J J J J J J J J J	EDU_DQH[3] KDU_DQH[3] K CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3]	100_CXE 100_ADCS[9] 1 100_ADCS[9] 100_CXE[9] 100_CXE[9] 100_CXE[9] 100_CXE[1] 100_CXE[1] 100_CXE[1]	H_LDDR(F) H SULPTION SULPTION (U_LD(10)	NDC_ADDR(6) NDC_ADDR(5) N NDC_ADDR(5) N NDC_ADDR(5) CPG_AD(13) CPG_AD(12) CPG_AD(12) CPG_AD(12)	MEX_ACOR[4] ACCC_ACOR[1] ACCC_A	REMINISTRATE OF THE PROPERTY O	MEDICAS JI T SECRETARISMENT SECRETARISMENT VEDS PLI_VED PLI_JYPASS	CPU_AD[14]	MENL DOM(1) MENL DATA(8) ***********************************	EDIL DATA (9) E E E E E E E E E E E E E	NEM. DATA [11] Y *********************************	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA CPU_VALIN_N CPU_AD(27) CPU_AD(26) CPU_AD(26)	IR_CLK 8 MISC_LED(1) MISC_LED(0) AB **********************************	EC_DATA[26]	AD	RIO_ADOR[20] RIO_ADOR[21] AE *********************************	RIO_DINT(1) AP STS_PAROK GPIO(9) GPIO(11) IO_DATA MEN_DATA(16) MEN_DATA(20) MEN_DATA(22) MEN_DOM(2) MEN_DOM(3) MEN_DOM(3)
A B C D E F G G H J K L M	: S: T:	STE_MGGK A STE_MGGK STE_MGGK STE_CLX STE_CL	GPIO(9) D SPIO(10) GPIO(10) GPIO	CPIO(11) C OFFICIALLY DIV_DATA(1) OFFIC(0) OFFIC(12) VDD SO_COM DAC_TYPE HO_COM DAC_COM VS22	DIV_DATA(4) DIV_DATA(4) DIV_DATA(2) VSS1 VSS1 VSS2 DAC_VREP DAC_CREBAOUT DAC_CREBAOUT	VSS2 MODE_DATA(16) MODE_DATA(1	224_DATA[17] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[17] NO P 224_DATA[18]	CPI_AD(5) DIV_SCATA DIV_SCATA DIV_SCATA	EDI_DATA[21] E EDI_DATA[22] E COLUMN (1) COLUMN (1)	J J J J J J J J J J J J J J J J J J J	EDU_DQH[3] KDU_DQH[3] K CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3]	100_CXE 100_ADCS[9] 1 100_ADCS[9] 100_CXE[9] 100_CXE[9] 100_CXE[9] 100_CXE[1] 100_CXE[1] 100_CXE[1]	H_LDDR(F) H SULPTION SULPTION (U_LD(10)	NDC_ADDR(6) NDC_ADDR(5) N NDC_ADDR(5) N NDC_ADDR(5) CPG_AD(13) CPG_AD(12) CPG_AD(12) CPG_AD(12)	MEX_ACOR[4] ACCC_ACOR[1] ACCC_A	REMINISTRATE OF THE PROPERTY O	MEDICAS JI T SECRETARISMENT SECRETARISMENT VEDS PLI_VED PLI_JYPASS	CPU_AD[14]	MENL DOM(1) MENL DATA(8) ***********************************	EDIL DATA (9) E E E E E E E E E E E E E	NEM. DATA [11] Y *********************************	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA CPU_VALIN_N CPU_AD(27) CPU_AD(26) CPU_AD(26)	IR_CLK 8 MISC_LED(1) MISC_LED(0) AB **********************************	EC_DATA[26] B VESS BESS BESS BESS BESS BESS BESS BESS	AD	RIO_ADDR(20) RIO_ADDR(21) AE *********************************	RIO_DIRT(1) AP STS_PARKE GPIO(9) GPIO(11) IO_DATA EXD_DATA(16) MEN_DATA(20) MEN_DATA(20) MEN_DATA(20) MEN_DATA(20) MEN_DATA(20) MEN_DATA(20) MEN_DATA(20) MEN_DATA(20) MEN_DATA(20)
	: 55 : Ti : DAV : VX	STE_MGGK A STE_MGGK STE_MGGK STE_CLX STE_CL	GPIO(9) D OPIO(9)	CPIO(11) C OFFICE OF THE STATE	DIV_DATA(4) DIV_DATA(4) DIV_DATA(2) VSS1 VSS1 DAC_YAOUT DAC_CREADOUT	VSS2 MODE_DATA(16) MODE_DATA(1	224_DATA[17] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[17] NO P 224_DATA[18]	CPI_AD(5) DIV_SCATA DIV_SCATA DIV_SCATA	EDI_DATA[21] E EDI_DATA[22] E COLUMN (1) COLUMN (1)	J J J J J J J J J J J J J J J J J J J	EDU_DQH[3] KDU_DQH[3] K CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3]	100_CXE 100_ADCS[9] 1 100_ADCS[9] 100_CXE[9] 100_CXE[9] 100_CXE[9] 100_CXE[1] 100_CXE[1] 100_CXE[1]	H_LDDR(F) H SULPTION SULPTION (U_LD(10)	NDC_ADDR(6) NDC_ADDR(5) N NDC_ADDR(5) N NDC_ADDR(5) CPG_AD(13) CPG_AD(12) CPG_AD(12) CPG_AD(12)	MEX_ACOR[4] ACCC_ACOR[1] ACCC_A	REMINISTRATE OF THE PROPERTY O	MEDICAS JI T SECRETARISMENT SECRETARISMENT VEDS PLI_VED PLI_JYPASS	CPU_AD[14]	MENL DOM(1) MENL DATA(8) ***********************************	EDIL DATA (9) E E E E E E E E E E E E E	NEM. DATA [11] Y *********************************	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA CPU_VALIN_N CPU_AD(27) CPU_AD(26) CPU_AD(26)	IR_CLK 8 MISC_LED(1) MISC_LED(0) AB **********************************	EDL_ADDR(19) R IR_IM R AC CPU_MOSSIM SEC_CLE VSS1 VSS2 VDD3 CPU_DATA(26) R VDD3 KEM_ADDR(2)	200_DEX_H[2] AD 200_DEX_H[2] AD 200_DEX_H[2]	RIO_ADDR(20) RIO_ADDR(21) AE *********************************	NO_CON RIO_DINT[1] AP RIO_DINT[1] AP RIO_DINT[1] RIO_CON RI
	: Si : Ti	STE_MGGK A STE_MGGK STE_MGGK STE_CLX STE_CL	GPIO(9) D GPIO(8) GPIO(8) GPIO(8) DIV_LIC STG_RENTCH_N HO_CON DAC_CONCEAPVO DAC_CONCEAPVO DAC_CONCEAPVO AC_CONCEAPVO VID_USTRIC_N VID_DATA(5) VID_DATA(1)	CPIO(11) C DIV_DATA(1) DIV_DATA(1) CPIO(2) CPIO(12) TEST_MOCE(6) WDD MC_CON DAC_CRCEVES MO_CON VSS2 VID_DATA(6) VID_DATA(6)	DIV_DATA(4) DIV_DATA(4) DIV_DATA(2) VSS1 VSS1 DAC_YAOUT DAC_CREADOUT	VSS2 MODE_DATA(16) MODE_DATA(1	224_DATA[17] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[17] NO P 224_DATA[18]	CPI_AD(5) DIV_SCATA DIV_SCATA DIV_SCATA	EDI_DATA[21] E EDI_DATA[22] E COLUMN (1) COLUMN (1)	J J J J J J J J J J J J J J J J J J J	EDU_DQH[3] KDU_DQH[3] K CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3]	100_CXE 100_ADCS[9] 1 100_ADCS[9] 100_CXE[9] 100_CXE[9] 100_CXE[9] 100_CXE[1] 100_CXE[1] 100_CXE[1]	H_LDDR(F) H SULPTION SULPTION (U_LD(10)	NDC_ADDR(6) NDC_ADDR(5) N NDC_ADDR(5) N NDC_ADDR(5) CPG_AD(13) CPG_AD(12) CPG_AD(12) CPG_AD(12)	MEX_ACOR[4] ACCC_ACOR[1] ACCC_A	REMINISTRATE OF THE PROPERTY O	MEDICAS JI T SECRETARISMENT SECRETARISMENT VEDS PLI_VED PLI_JYPASS	CPU_AD[14]	MENL DOM(1) MENL DATA(8) ***********************************	EDIL DATA (9) E E E E E E E E E E E E E	NEM. DATA [11] Y *********************************	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA CPU_VALIN_N CPU_AD(27) CPU_AD(26) CPU_AD(26)	IR_CLK 8 MISC_LED(1) MISC_LED(0) AB **********************************	EC_DATA[26] B VESS BESS BESS BESS BESS BESS BESS BESS	200_DESC_H[2] AD 200_DESC_H[2] AD 200_DESC_H[2]	RIO_ADDR(20) RIO_ADDR(21) AE *********************************	RIO_DINT[1] AP RIO_DINT[1] AP RIO_DINT[1] AP RIO_DINT[1] RIO_DINT[1] RIO_DATA[16] RIO_DATA[16] RIO_DATA[20] RIO_DATA[20] RIO_DATA[20] RIO_DATA[20] RIO_DATA[20] RIO_DATA[20] RIO_DATA[20] RIO_DON[2] RIO_DON[3] RIO_LONE[5] RIO_LONE[5] RIO_LONE[5] RIO_LONE[5]
	: Si : Ti	TIC_CLX TIC_CLX TIC_CLX TIC_DATA TIC_DATA TIC_COND	GPIO(9) D GPIO(8) GPIO(8) GPIO(8) DIV_LIC STG_RENTCH_N HO_CON DAC_CONCEAPVO DAC_CONCEAPVO DAC_CONCEAPVO AC_CONCEAPVO VID_USTRIC_N VID_DATA(5) VID_DATA(1)	CPIO(11) C DIV_DATA(1) DIV_DATA(0) CPIO(12) TEST_MOCE(0) MO_COM MO_COM MO_COM VSS2 VIO_DATA(6) VIO_DATA(6) CPIO(12) CPIO(12) CPIO(12) CPIO(12) CPIO(12) CPIO(12) CPIO(12) CPIO(13)	DIV_DATA(4) DIV_DATA(4) DIV_DATA(2) VSS1 VSS1 VSS2 DAC_VACOT DAC_VACOT DAC_CCGAOOT DAC_CCGAOOT VID_DATA(7) VID_DATA(4)	VSS2 MSC_DATA(16)	224_DATA[17] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[18] NO P 224_DATA[17] NO P 224_DATA[18]	CPI_AD(5) DIV_SCATA DIV_SCATA DIV_SCATA	EDI_DATA[21] E EDI_DATA[22] E COLUMN (1) COLUMN (1)	J J J J J J J J J J J J J J J J J J J	EDU_DQH[3] KDU_DQH[3] K CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3] CHU_DQH[3]	100_CXE 100_ADCS[9] 1 100_ADCS[9] 100_CXE[9] 100_CXE[9] 100_CXE[9] 100_CXE[1] 100_CXE[1] 100_CXE[1]	H_LDDR(F) H SULPTION SULPTION (U_LD(10)	NDC_ADDR(6) NDC_ADDR(5) N NDC_ADDR(5) N NDC_ADDR(5) CPG_AD(13) CPG_AD(12) CPG_AD(12) CPG_AD(12)	MEX_ACOR[4] ACCC_ACOR[1] ACCC_A	REMINISTRATE OF THE PROPERTY O	MEDICAS JI T SECRETARISMENT SECRETARISMENT VEDS PLI_VED PLI_JYPASS	CPU_AD[14]	MENL DOM(1) MENL DATA(8) ***********************************	EDIL DATA (9) E E E E E E E E E E E E E	NEM. DATA [11] Y *********************************	MEN_DATA(0) MEN_DATA(13) MEN_DATA(14) AA CPU_VALIN_N CPU_AD(27) CPU_AD(26) CPU_AD(26)	IR_CLK 8 MISC_LED(1) MISC_LED(0) AB **********************************	ED_ADDR(19) # IR_IM # AC CPU_AD[22] C CPU_MODEIM SMC_CLX VSS1 VSS1 VSS2 E VDD0 8 VDD0 8 MEH_ADDR(2) VSD2 VSD3	AD	RIO_ADDR(20) RIO_ADDR(21) AE *********************************	RIO_DINT[1] AP SYS_PARKE GPIO(9) GPIO(11] ID_DATA MEN_DATA(16) MEN_DATA(20) MEN_DOM(2) MEN_DOM(3) MEN_DOM(9) 2021_ADDR(7) MEN_ADDR(1) MEN_ADDR(1) MEN_ADDR(10)

| NO_STATE | NO_STATE

VSS3

VSS1 VD03

VDO VSS1 VSS1 VSS1

UART_TED UART_CTS_H UART_RED HOD_SDATAIN RIQ_EXPEN RIO_DINT(0) RIO_MER_H RIO_DATA(20) RIO_DATA(26) RIO_DATA(25) RIO_DATA(21) RIO_ADDR(1) RIO_ADDR(1) RIO_ADDR(1) VSS1 VSS1

HOD_CLE RIO_EXPACE_N RIO_CR_N(1) RIO_CR_N(0) VSSI RIO_DATA(21) VDD RIO_DATA(24) RIO_DDR(16] RIO_ADDR(16) VDD RIO_ADDR(11)

PP_ACE_H PP_EDSY PP_ERROR CPTO[14]

PP_DIR GPIO(6) PP_SELECT

VDD3 HERCDATA(5) HERCDON(0) HERCES_N(0)

VSS1 MEM_DATA(2) MEM_DATA(9) MEM_DATA(10)
VSS1 MEM_DATA(1) MEM_DATA(11) MEM_DATA(12)

VSS2 MEM_DATA(4) MEM_DATA(3) MEM_DOM(1) MEM_DATA(8)

VSS2 MENLDATA(15) MENLDATA(0) MENLDATA(13) MENLDATA(14)

NO_CON VSS1 IR_CLK MISC_LED[1] MISC_LED[0]

VSS1 VSS1 VSS1 VSS1 RIO_DATA[8] RIO_ADDR[11] RIO_DATA[15] RIO_ADDR[10] RIO_DER_H[1] RIO_DER_H[2]

VSS1 GPIO[15] RIO_ACOR[19] HISC_LED[2] IR_IN

SOLO1 Rev. 1.3 Errata

A.10verview

This appendix describes the changes that comprise Revision 1.3 of the SOLO1 ASIC.

A.2 SDRAM Self-refresh Fix

The Auto Refresh mode has been fixed in SOLO1 Rev 1.3. Auto Refresh is a mode that allows the SDRAM chips to refresh themselves without external stimulus from SOLO1. If Auto Refresh is enabled, SOLO1 will place the SDRAMs in autorefresh mode automatically, if no memory requests are presented to the memUnit between normal refresh cycles. To enable this feature, software must write 0xF0000411 (66.7 Mhz) or 0xF0000516 (83.3Mhz) to the MEM_REFCTRL Register.

A.3 External Master/Slave Device on CPU Interface

SOLO1 Rev 1.3 provides an enhancement to the CPU bus interface that allows a master/slave device to share the bus with the CPU and SOLO1. The supported protocol allows the device to act as a master to SOLO1 and as either a master or a slave to the CPU. However, currently supported CPUs have no slave resources and hence, external devices are limited to acting as a slave to the CPU and a master to SOLO1.

A.3.1 Enabling the Enhanced CPU Interface

The enhanced-CPU interface is enabled by providing a pullup resistor on the system PCB for RIO_ADDR[4]. The RIO_ADDR bus is connected to the configuration resistors used to set various chip and system configurations. Each bit of the RIO_ADDR bus (bits 22..2) must be connected to either a pullup or pulldown resistor on the PCB.

In previous versions of SOLO1, the reset state of RIO_ADDR[4] was readable by software, but did not affect the hardware functionality. In SOLO1 Rev. 1.3, if RIO_ADDR[4] is pulled down, the CPU interface is compatible with previous versions of SOLO1. This mode should be used when no external master/slave device is provided in the system. If RIO_ADDR[4] is pulled up, the enhanced-CPU interface is enabled. This mode should be used when an external master/slave device is provided in the system.

A.3.2 Solo and External Device Address Spaces

Transactions on the CPU bus are designated for either SOLO1, or for the external device, based on their addresses. When the old, compatible CPU interface is implemented, all transactions on the CPU bus are handled by SOLO1. If the transaction's address is not defined in SOLO1's address map, a bus error is generated. When SOLO1 is implementing the new, enhanced-CPU interface, transactions whose addresses have bit 31 set are ignored by SOLO1 and assumed to be handled by the external device. Transactions whose addresses do not have bit 31 set are handled by SOLO1.

The system address maps for compatible and enhanced CPU interface-based systems are shown in Figure A-1.

Compatible CPU Interfac		ce E	inhanced CPU Interface
0x 		0xffff_ffff	External Device Address Space
0x8000_0000 0x2000_0000	SOLO/Reserved	0x8000_0000 0x2000_0000	SOLO/Reserved
0x1f00_0000	SOLO/ROM	0x1f00_0000	SOLO/ROM
0x1d00_0000	SOLO/Devices	0x1d00_0000	SOLO/Devices
0x0800_0000	SOLO/Reserved	0x0800_0000	SOLO/Reserved
0x0480_0000	SOLO/Exp Ports	0x0480_0000	SOLO/Exp Ports
0x0400_0000	SOLO/Control Regs	0x0400_0000	SOLO/Control Regs
0x0000_0000	SOLO/RAM	0x0000_0000	SOLO/RAM

Figure A-1 System Address Maps

Transactions to addresses which fall into the spaces identified as "SOLO/Reserved" will cause SOLO1 to generate a bus error. The external device is responsible for generating bus errors for transactions to undefined addresses within its address space.

Note that the r4640 CPU cannot generate addresses in the 0x2000_0000 - 0xffff_ffff range. If an r4640 is used in a system that implements the enhanced-CPU interface, and has an external device, the CPU cannot access that external device. In such a system, both the CPU and the external device can act as masters of SOLO1, but cannot communicate with each other in any way. The QED-5230 has an MMU and can generate transactions across the full range of addresses from 0x0000_0000 - 0xffff_ffff. Hence, it is the more likely candidate for systems taking advantage of the enhanced-CPU interface.

A.3.3 Enhanced Mode CPU Interface Signals

Most signals comprising the CPU interface are shared between SOLO1, the CPU and an external device (when all three are present in a system). Two additional inputs to SOLO1 (both driven by the external device) are required to implement the enhanced interface— EXT_VALIDOUT_N and EXT_WRRDY_N.

EXT_VALIDOUT_N is analogous to the CPU_VALIDOUT_N signal driven by the CPU. When the external device is acting as the bus master, it asserts EXT_VALIDOUT_N on cycles during which it is driving valid address or data information on the CPU_AD and CPU_CMD buses. SOLO1 Rev. 1.3 uses the TEST_SCANEN pin to receive the EXT_VALIDOUT_N signal during enhanced-mode operation.

EXT_WRRDY_N is analogous to the CPU_WRRDY_N signal driven by SOLO1. When the external device is capable of accepting a write transaction from the CPU, it asserts EXT_WRRDY_N. SOLO1 combines the state of this signal with it's own internal state to determine the value of CPU_WRRDY_N. Since the target of the next CPU transaction cannot be guessed ahead of time, if either of the two slaves are not ready to accept a write transaction, then CPU_WRRDY_N is deasserted. Both the CPU and the external device must respect the state of CPU_WRRDY_N when determining the state of a write transaction being presented on the bus. The external device uses EXT_WRRDY_N to control the state of CPU_WRRDY_N, but must still respect the state of CPU_WRRDY_N regardless of whether EXT_WRRDY_N has been deasserted. SOLO1 Rev. 1.3 uses the GPIO[2] pin to receive the EXT_WRRDY_N signal during enhanced-mode operation.

The behavior of the CPU_VALIDIN_N signal, driven by SOLO1, is modified for enhanced-mode operation. In compatibility mode, this signal is always driven by SOLO1. In enhanced mode, this signal is normally tri-stated and is driven by SOLO1 only when returning read data. After returning the appropriate data, SOLO1 drives CPU_VALIDIN_N high for one cycle before tri-stating the signal (CPU_VALIDIN_N must be weakly pulled up on the PCB to ensure that the deasserted high value is held while the signal isn't being driven). The external device must follow this same protocol for driving CPU_VALIDIN_N: it must be tri-stated when the device is not the bus master and must be driven high (deasserted) for at least one cycle prior to being tri-stated again. The external device uses CPU_VALIDIN_N for returning read data to the CPU, but also uses it to assert a null transaction to the CPU— returning mastership of the bus to the CPU, following a series of transactions issued by the external device to SOLO1.

A.3.4 Enhanced Mode System Description

Figure A-2 shows the required connections between the CPU, SOLO1 and an external device in a typical system configuration. Note that the CPU_MODECLK, CPU_MODEIN and the various reset signals are not shown. These signals are

typically connected between the CPU and SOLO1 as previously described. Whether the external device needs to be connected to any of these signals is a system decision. Typically, the external device is connected to at least one reset signal, although whether that signal is one of the CPU reset signals or a more generic reset signal, used for other system devices, is not specified here.

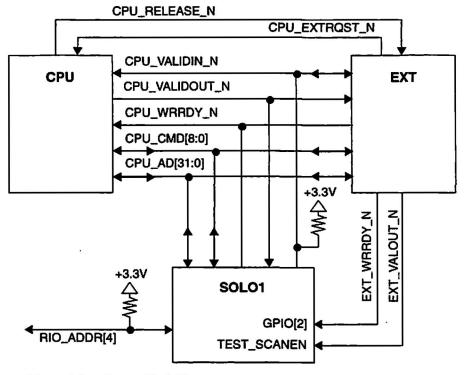


Figure A-2 System Block Diagram

As previously discussed, the external device drives two new signals to SOLO1. They are connected to SOLO1's GPIO[2] and TEST_SCANEN pins. The external device also drives the CPU_EXTRQST_N signal, which is tied high in sytems with only SOLO1 and the CPU. CPU_RELEASE_N, which is otherwise left unconnected, is read by the external device to determine when the CPU is responding to an asserted CPU_EXTRQST_N signal.

A.3.5 Transactions with the CPU as Master

Transactions where the CPU acts as the bus master are handled by both SOLO1 and the external device in the same way: The CPU issues a transaction which is recognized by both SOLO1 and the external device. If bit 31 of the transaction address is set, the external device responds, if bit 31 is not set, SOLO1 responds. Write transactions are performed by the appropriate target, although no external

response is generated—except for a possible deassertion of WRRDY_N signals while the write target clears its write buffer. Read transactions are handled by the appropriate target, which begins driving CPU_AD, CPU_CMD and CPU_VALIDIN_N once the CPU transitions to slave state. After read data is returned, the target device deasserts CPU_VALIDIN_N for one cycle and tristates CPU_AD and CPU_CMD. On the following cycle, the target device tristates CPU_VALIDIN_N as the CPU becomes the bus master once again.

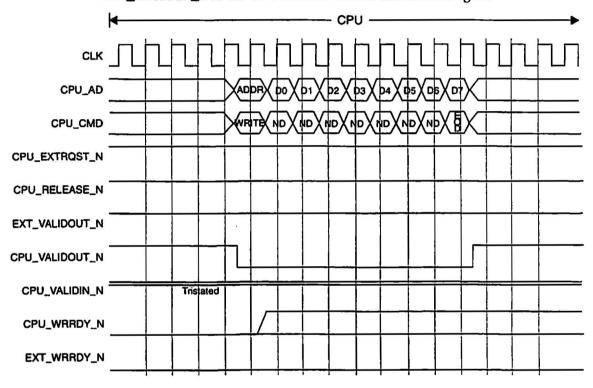


Figure A-3 CPU Write to SOLO1

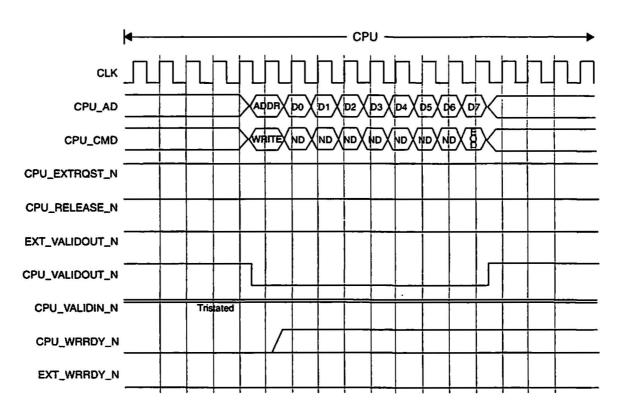


Figure A-4 CPU Write to an External Device

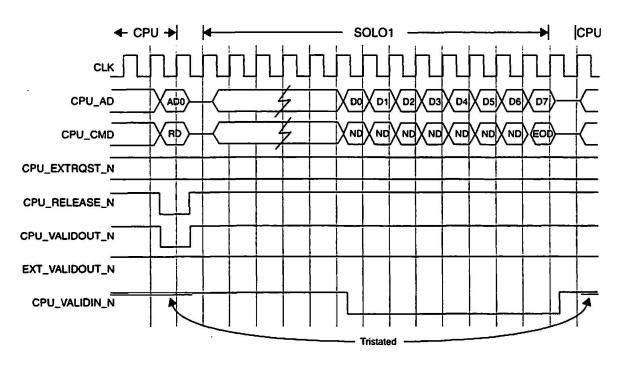


Figure A-5 CPU Read from SOLO1

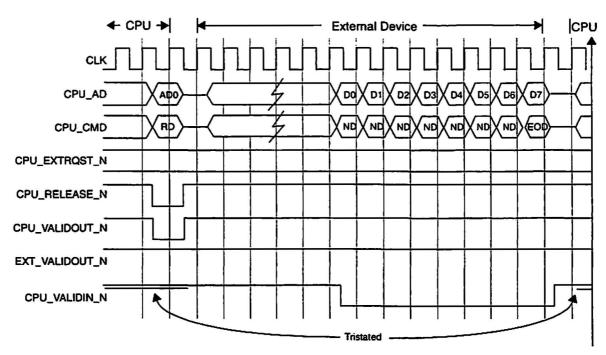


Figure A-6 CPU Read from an External Device

A.3.6 Transactions with External Device as Master

The external device uses the CPU_EXTRQST_N signal to indicate to the CPU that it wants to become the bus master. The CPU responds by asserting CPU_RELEASE_N and deasserting CPU_VALIDOUT_N one cycle before transitioning to the slave state. After the CPU asserts CPU_RELEASE_N, the bus must be tristated for one cycle, after which the external device becomes the bus master.

Once it is the bus master, the external device can issue read and write transactions to SOLO1 using the EXT_VALIDOUT_N signal, instead of the CPU_VALIDOUT_N signal used by the CPU. Note that the CPU has no slave resources, so the external device cannot issue transactions to it.

After completing its transactions, the external device must return the bus to the CPU. This is done by sending a null transaction request to the CPU on the CPU_CMD lines, while asserting CPU_VALIDIN_N. After the device drives the null transaction, the bus must be tristated for one cycle, after which the CPU becomes the bus master.

See the following figures for the timing associated with reads and writes to SOLO1.

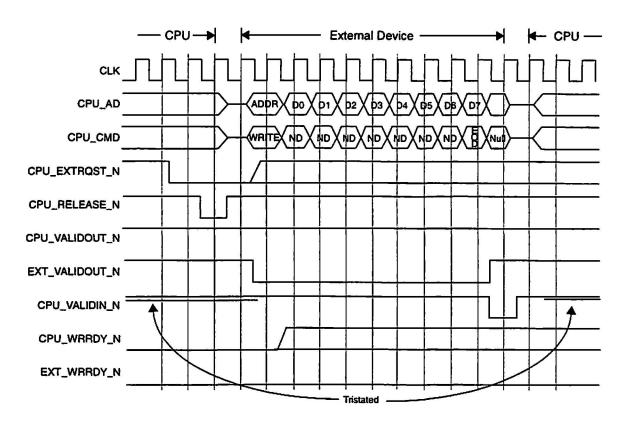


Figure A-7 External Device Write to SOLO1

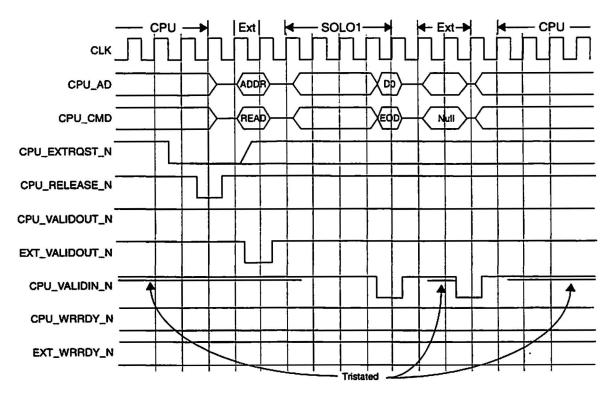


Figure A-8 External Device Single-word Read from SOLO1

While the external device is the bus master, the CPU is waiting for the null transaction which will transfer bus mastership. Cycles during which SOLO1 is returning read data to the external device are ignored by the CPU even though SOLO1 is asserting CPU_VALIDIN_N.

A.3.7 Transactions Issued While Returning Read Data to the CPU

Because the CPU has given up mastership of the bus, the external device has the opportunity to issue write transactions to SOLO1 before returning read data to the CPU in response to a read request. Read transactions to SOLO1 cannot be inserted this way because the CPU_VALIDIN_N signal is shared and would cause the CPU to accept the read response data from SOLO1, instead of waiting for the appropriate data to be returned from the external device.

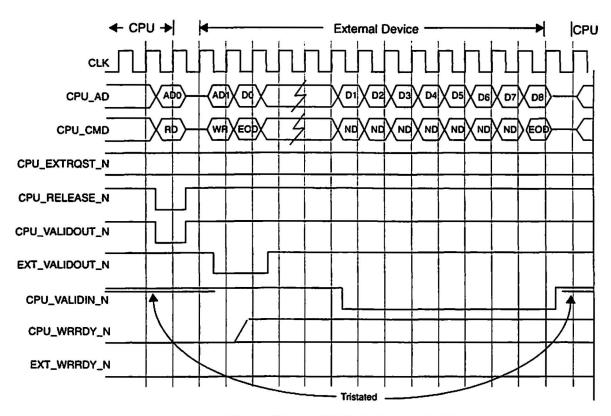


Figure A-9 External Device Write to SOLO1 Underneath Read Response

A.3.8 Transaction Size Restrictions

There have been no modifications to support transaction sizes other than those used by the CPU. Hence, transactions are limited to 32-byte blocks or 4-byte single words. Block transactions must contain 32 bytes of valid data, while word transactions can specify byte, half-word or tri-byte masks.

The practical restrictions imposed by these limitations do not substantially affect read behavior. Read transactions of varying sizes can be accomplished by issuing a block read and ignoring the unwanted data. Note, that if the device requires access to SOLO1 control registers, it must also implement single-word read transactions, as the control registers cannot be accessed with block transactions. RAM, ROM and WebTV Port expansion devices may all be accessed with block read transactions.

Write behavior is more adversely affected by the imposed transaction size limitations. If writes can be collected into complete, aligned, 32-byte transactions, then they can take advantage of block transfers. Otherwise, write data must be broken into multiple single-word transfers, each of which may use a byte, half-word or tri-byte mask. Again, SOLO1 control registers may only be accessed through single-word transactions. Furthermore, ROM resources, while supporting block reads, do not support block write transfers. RAM and WebTV Port expansion devices may be accessed with block write transactions.

A.3.9 Transaction Performance Issues

Note, that since SOLO1's RAM memory interface is designed to prefer 32-byte accesses, there are some non-obvious performance issues which arise. These issues are the result of both the memory controller's preference for 32-byte transactions, as well as the CPU interface's ability to deal with this preference.

First, the underlying number of cycles required to complete a memory access is the same, regardless of whether the access is for a single word or a 32-byte block. Including both the underlying data transfer time, as well as the overhead of the transactions, means that transferring 32 bytes of data with eight single-word accesses is eight times slower than using a single block transaction. Hence, there is a very strong performance preference for using block transfers wherever possible.

Futhermore, for block read transactions, SOLO1 adheres to a policy of "critical word first." Single-word reads are accomplished by performing a 32-byte read from RAM and ignoring the data belonging to the seven words not requested.

The single word belonging to the read transaction is always returned as the last four bytes of the 32 bytes transferred from RAM. So, single-word reads are more efficiently accomplished as block transfers, because the desired word will be returned seven cycles ahead of when it would have been returned by the equivalent single-word read transaction. This enhancement requires that the external device be capable of ignoring the seven additional data cycles generated by a block read request. If the device is not capable of this, single-word reads can be issued at the expense of the additional seven-cycle latency.

The performance of single-word write transactions is helped somewhat by the presence of a write buffer in SOLO1. However, each individual transaction still requires the same underlying data transfer time as a 32-byte block write. Unfortunately, all 32 bytes of a block write must be valid, so there is no easy way to accomplish transfers of fewer than 32 bytes without resorting to the much slower single-word transactions.

One approach would be to assemble the data to be written in a 32-byte write buffer, keeping track of which bytes are valid bytes to be written. Then, rather than breaking up the buffer into multiple single-word writes, the appropriate block read can be performed to fill in the correct values for bytes which are not intended to be written. Having filled the entire 32-byte buffer with valid data, a single block write transaction can be used. This scheme would result in the use of two block transactions and would be beneficial when at least three single-word transactions would otherwise have been used.

SOLO1 Errata Summary

This appendix lists all of the known problems with the SOLO1 chip and the status of those problems.

Table B-1 Summary of SOLO1 Problems

Description of Bug	Rev 1.0	Rev 1.1	Rev 1.1.1	Rev 1.3
Chip rev value (BUS_CHIPID[23:20])	0x0	0x1	0x1	0x3
Video Output:				
Video output timing incorrect due to an extra clock	NF	F	F	F
DAC input hold timing violation	NF	F	F	F
Garbage video on right edge of screen	NF	F	F	F
DVE horizontal filter doesn't work	NF	NF	NF	NF
Field interrupts should be driven by sync edges, not state	NF	NF	NF	NF
Some YUV values may cause over/under flow	NF	NF	NF	NF
shiftInt is in POT_INTEN but not in POT_INTSTAT	NF	NF	NF	NF
Odd length frames not handled correctly	NF	NF	NF	NF
GFX RAM pre-charge timing parameter not met	NF	F	F	F
Extra lines at top of screen during GFX writeback	NF	NF	NF	NF
deltaTim mode broken for all VQ modes	NF	NF	NF	NF
ping-pong out of time occasionally hangs gfxUnit	NF	NF	NF	NF
Write-back should calculate effective FB start on even fields	NF	NF	NF	NF
Random pixels dropped in constant write-back mode	NF	NF	NF	NF

Table B-1 Summary of SOLO1 Problems (Continued)

Description of Bug	Rev 1.0	Rev 1.1	Rev 1.1.1	Rev 1.3
deltaTim mode does not work with microcels	NF	NF	NF	NF
Can't do pingpong mode with fullscreen video in PAL	NF	NF	NF	NF
Video Input:			······································	
divUnit dma chaining not working as specified	NF	NF	NF	NF
divUnit audio doesn't stop cleanly	NF	NF	NF	NF
divit decimation first column wrong at slow sys clock speeds	NF	NF	NF	NF
dev_GPIO[8] doesn't drive low when used as pwm	NF	NF	NF	NF
Memory:				
Memory self-refresh doesn't work	NF	NF	NF	F
MemUnit cannot address contiguous 8MBs of memory with x8 parts	NF	NF	NF	NF
memUnit REFENABLE register not correct	NF	NF	NF	NF
Test:				
PLL test mode doesn't tri-state VID_DATA bus	NF	NF	NF	NF
Need to disable disp_CbCrSel in frlbWrap during RAM test	NF	NF	NF	NF
Other:		**************************************		
PLL has excessive jitter	NF	NF	F(1)	F
GPIO[11] shouldn't have pullup	NF	F	F	F
aud/mod write request address bit 4 should not be used	NF	F	F	F
Parallel port address aliasing	NF	F	F	F
Can't access expansion devices 4 through 7	NF	NF	NF	NF
busUnit ERRADDR register not updated correctly	NF	NF.	NF	NF
Sense of the CTS_N flow control is inverted	NF	NF	NF	NF
Support for additional master/slave devices on CPU bus	NF	NF	NF	F
SOLO1.3 freezes on writes to addresses with bit 31 set.	_	-		NF

NF = Not Fixed

F = Fixed

(1) See section B-1 "Errata Details," for the Rev. 1.1.1 fix.

B.1 Errata Details

B.1.1 Video Output

Summary Number 1: SOLO1 video timing incorrect due to extra clock.

BugFlash #: N/A

Fixed in: Rev. 1.1

Details: The timing of the video signal coming out of the on-chip video encoder is incorrect due to an extra clock cycle being generated by the DVE.

Workaround: None

Summary Number 2: DAC input hold timing violation.

BugFlash #: N/A

Fixed in: Rev. 1.1

Details: The timing at the input of the on-chip video DACs does not meet hold

time.

Workaround: None

Summary Number 3: Garbage video on right edge of screen.

BugFlash #: N/A

Fixed in: Rev. 1.1

Details: Garbage video on right edge of screen.

Workaround: None

Summary Number 4: DVE horizontal filter doesn't work.

BugFlash #: 030512

Fixed in: Not fixed

Details: The notch filter is missing one coefficient bit, and there are problems with

overflow.

Workaround: Do horizontal filtering in software.

Summary Number 5: Field interrupts should be driven by sync edges, not state.

BugFlash #: 018563

Fixed in: Not fixed

Details: Currently, the potUnit detects the edge of the oddField signal to generate interrupts. If we get successive odd fields, either by time compression or a progressive DENC, we'll never see those edges. Ensure that no other internal state is based on the oddField state. (Signal to divUnit?)

Workaround: None

Summary Number 6: Some YUV values may cause over/under flow.

BugFlash #: 018900

Fixed in: Not fixed

Details: All values of YUV that can be generated from RGB are OK. But other values of YUV, which are legal, can cause under/over flow in the output caculations. These need to be clipped.

Workaround: None

Summary Number 7: shiftInt is in POT_INTEN register, but not in POT_INTSTAT register.

BugFlash #: 019479

Fixed in: Not fixed

Details: The shiftInt interrupt has a field in the POT_INTEN interrupt enable register, but it doesn't have an entry in the POT_INTSTAT interrupt status register. It does get correctly OR'd into pot_Int, however.

Workaround: When software detects a pot_Int and none of the POT_INTSTAT bits are set to assume that it's a shiftInt.

Summary Number 8: Odd length frames not handled correctly.

BugFlash #: 018204

Fixed in: Not fixed

Details: pot_vsize is in frame lines, and thus can be an odd number. However, we do a simple /2 and compare with field lines to see if we've hit the end. This will fail if the screen has an odd number of frame lines.

Workaround: None

B.1.2 Graphics

Summary Number 1: GFX RAM pre-charge timing parameter not met.

BugFlash #: 020669

Fixed in: Rev. 1.1

Details: For all 4 RAMs, Motive shows us that the AEN pulse width high parameter has not been met at worst-case conditions. We miss by approx. 0.4 ns. (However, it should operate at frequency in nominal conditions.)

Workaround: None

Summary Number 2: Extra lines at top of screen during GFX write-back.

BugFlash #: N/A

Fixed in: Not fixed

Details: None

Workaround: Change the vidUnit to start the frame buffer two lines later (i.e. at the start of line three), and thus miss the incorrect data. You'll also have to shorten the potUnit's VSize by two, and the vidUnit's NSize by 2*2*HSize.

Summary Number 3: deltaTim mode broken for all VQ modes.

BugFlash #: 021309

Fixed in: Not fixed

Details: The deltaTim mode (used for better accuracy on edges of cels) does not work on VQ textures. In fact, it hangs the gfx system, which in turn hangs the write-back unit. (To get out of the hang, try using the soft reset bit in the gfx control register.) The error was found in fctl_adr.cv in the RDST_IDLE state.

Workaround: None

Summary Number 4: Ping-pong out of time occasionally hangs gfxUnit

BugFlash #: 021622

Fixed in: Not fixed

Details: In ping-poing mode, when displaying a screen containing a large number of scan lines that run out of time, the entire screen will go black after some random amount of time. It does not come back until I switch out of ping-pong. I'm guessing it's related to out-of-time scan lines, because the problem does not show up with smaller images (like the WebTV logo), but does show up with the 444 color ramp.

To see the problem in the GFX test world, type the following at the grfx prompt:

i 1

t 1

start

Once the screen goes black, you can recover by typing:

wb

pp

YMaps subroutines are on by default. To turn them off, use the "sub"command. The screen black-out will happen much less frequently once subroutines are off.

Workaround: None

Summary Number 5: Write-back should calculate the effective frame buffer start on even fields.

BugFlash #: 018146

Fixed in: Not fixed

Details: Currently, the starting address for the frame buffer has to be changed by software. When the evenEnable bit gets hit, the hardware should calculate the new first line based off the start of the odd field.

Workaround: None

Summary Number 6: Random pixels dropped in constant write-back mode.

BugFlash #: 018063

Fixed in: Not fixed

Details: If you write a test that composes a complex image, write-back once and then continuously display, the image looks fine. If you constantly write-back then random sparkling bad pixels show up across the screen.

Workaround: None

Summary Number 7: deltaTim mode does not work with microcels.

BugFlash #: 021623

Fixed in: Not fixed

Details: It looks like deltaTim doesn't work with MiniCelRecords, even if the texture is direct 444 or 422. It appears to repeat the same line of pixels on each scanline.

Workaround: None

Summary Number 8: Can't do ping-pong mode with fullscreen video in PAL.

BugFlash #: 021147

Fixed in: Not fixed

Details: Line buffers are only 640. Fullscreen video in PAL requires 768. This could cause major differences in browser functionality between the NTSC and PAL versions.

Workaround: None

B.1.3 Video Input

Summary Number 1: divUnit DMA chaining not working as specified.

BugFlash #: 019911

Fixed in: Not fixed

Details: Setting the nextValid bit alone doesn't work. Either nextValidForever or ping-pong modes must be set. To perform normal chaining, use nextValidForever and be sure to clear it if you ever need to shut down the DMA. This is true for both video and audio.

Workaround: None

Summary Number 2: divUnit audio doesn't stop cleanly.

BugFlash #: 019912

Fixed in: Not fixed

Details: divUnit's audio wasn't designed to cleanly stop on a desired capture buffer.

Workaround: The workaround is to submit an additional buffer and clear the audCaptureEnable bit once the previous buffer's interrupt is received.

Summary Number 3: Divit decimation first column is wrong at slow system clock speeds.

BugFlash #: 021609

Fixed in: Not fixed

Details: If the system clock speed is less than 2x the decoder clock speed, then the leftmost column of decimated images can be wrong. The fastest decoder clock is 36MHz with PAL and the system clock speed is nominally 83MHz, so this should not be a problem. The bug is due to the fact that the decPhase bit is not cleared at the end of every line.

Workaround: None

Summary Number 4: dev_GPIO[8] doesn't drive low when used as pwm.

BugFlash #: 021672

Fixed in: Not fixed

Details: dev_GPIO[8] should pass through the div_pwm signal if div_pwmEnable is high. However, dev_gpio_en_nx[8] is also assigned to div_pwm if div_pwmEnable is high. This means that when this GPIO is used for pwm, it will drive 1's, but float when it should drive 0's.

Workaround: This can be fixed with a pulldown of the right strength on the board.

B.1.4 Memory

Summary Number 1: Memory self-refresh doesn't work.

BugFlash #: 021035

Fixed in: Rev. 1.3

Details: CKE timing to the SDRAM is incorrect.

Workaround: Toshiba SDRAMs seem to tolerate this timing protocol.

Summary Number 2: memUnit cannot address 8MBs of contiguous memory with x8 parts.

BugFlash #: 021035

Fixed in: Not fixed

Details: Address bit 23 is not routed properly for 16Mbit parts, meaning that the 8MBs of SDRAM must be accessed in two separate 4-MB chunks. This problem would not exist with 16Mbit parts, but would require a board spin.

Workaround: Don't use x8 parts.

Summary Number 3: memUnit REFENABLE register not correct.

BugFlash #: 021391

Fixed in: Not fixed

Details: In the memUnit RefEnable Register, the SetRefEnb and setAutoRef enables must be set to *set* the ref enable or auto refEnable, but writing a "0" to ref enable or auto Ref Enable will clear the values.

Workaround: Software must keep a mask to avoid clearing these bits unintentionally.

B.1.5 Test

Summary Number 1: PLL test mode doesn't tri-state the VID_DATA bus.

BugFlash #: 020754

Fixed in: Not fixed

Details: When in PLL TEST MODE, the vidUnit continues to drive it's output signals such that it's impossible to drive the PLL signals to test the PLL properly.

Workaround: Perform a write to the potUnit to disable the vidUnit.

Summary Number 2: Need to disable disp_CbCrSel in frlbWrap during RAM test.

BugFlash #: 019709

Fixed in: Not fixed

Details: It would have been nice to disable the way the fvo side of the RAMs rely on disp_CbCrSel to be active in order to do a read/write.

Workaround: Take disp_CbCrSel timing into account, and stimulate HSYNC/ VSYNC correctly to get CbCrSel running.

B.1.6 Other

Summary Number 1: PLL has excessive jitter

BugFlash #: N/A

Fixed in: Rev. 1.1.1, Rev. 1.3

Details: Rev. 1.0— PLL had a layout problem that prevented operation at normal voltages. Rev. 1.1—PLL fixed the voltage sensitivity, but had excessive jitter when supplied with the spread-spectrum clock. Rev. 1.1.1—PLL improved, but still outside of the clock skew budget. Rev. 1.3—Significant layout changes to improve the jitter.

Workaround: Rev. 1.0—Use a voltage regulator on VAA to keep the voltage at 2.7V. Rev. 1.1—Use a voltage regulator on VAA to keep the voltage at 3.3V. Rev. 1.1.1— Use a voltage regulator on VAA to keep the voltage at 3.3V.

Summary Number 2: GPIO[11] shouldn't have a pullup.

BugFlash #: N/A

Fixed in: Rev. 1.1

Details: GPIO[11] is used as a "Power OK" signal, which tells SOLO1 that it is safe to issue the system reset. Since an external RC network may be used to control the timing of this signal, an internal pullup is inappropriate.

Workaround: Consider the value of the internal pullup when calculating the external RC values.

Summary Number 3: Aud/mod write request address bit 4 should not be used.

BugFlash #: N/A

Fixed in: Rev. 1.1

Details: The audUnit and modUnit perform writes in 4-word bursts; thus, the DQM bits are used. The memUnit assumes that the write address is 8-word aligned; if address bit 4 is a '1', the write may not complete and the memUnit may hang.

Workaround: None

Summary Number 4: Parallel Port address aliasing.

BugFlash #: 029621

Fixed in: Rev. 1.1

Details: The Parallel Port registers in SOLO1 alias to other registers in the devUnit, because the address is not fully decoded. They should be fully decoded in devPPort.rv.

Workaround: None

Summary Number 5: Can't access expansion devices 4 through 7.

BugFlash #: 020755

Fixed in: Not fixed

Details: The row address sent to expansion devices 4-7 is not generated correctly. This is only a problem if more than 3 expansion devices are plugged in at one time.

Workaround: There is a software workaround which would apply to accesses to devices 4-7 only.

Summary Number 6: busUnit ERRADDR register is not updated correctly.

BugFlash #: 029369

Fixed in: Not fixed

Details: The ERRADDR register, in the busUnit, reports addresses which generate bus errors or violate any fences. Fences are not enabled/disabled except through the interrupt enable registers. This means that even if the interrupt is disabled, the fence is still "active" and hence, when one of the fences is used to write protect code space in RAM, every code fetch generates a read violation which doesn't generate an interrupt, but overwrites the ERRADDR register.

Workaround: None

Summary Number 7: Support for additional master/slave devices on the CPU

BugFlash #: N/A

Fixed in: Rev. 1.3

Details: Rev. 1.3 adds support for an additional master/slave device on the CPU bus. Refer to Appendix A, "SOLO1 Rev. 1.3 Errata."

Workaround: None

Summary Number 8: Sense of the CTS_N flow control is inverted.

BugFlash #: 40768

Fixed in: Not fixed

Details: There is a hardware flow control mechanism for the sucUnit serial port. When CTS_N is asserted (i.e. low), it should enable the transmit FIFO to send data. When it is deasserted (i.e. high), it should disable the FIFO after the current character is sent. In all implementations of SOLO1 so far, the sense of the CTS_N flow control is inverted (i.e. it will enable the FIFO when CTS_N is high). Since most other serial devices would be confused by this behavior, SOLO1 has very little chance of using this mode effectively.

Workaround: There is a software workaround for this in that software will take an interrupt and disable the FIFO. This is the way that the 16550 works anyway, so it should not be a big deal.

The hardware workaround would be to put an inversion at the CTS_N SOLO1 input, but this is not really good since software would have to know that this inverter is actually there. Plus the CTS_N signal would be inverted for the cases where hardware flow control is not used.

Summary Number 9: SOLO1.3 freezes on writes to addresses with bit 31 set.

BugFlash #: N/A

Fixed in: SOLO2

Details: The CPU interface compatibility mode in SOLO1.3 contains a bug that halts the CPU bus if an external write transaction is issued to the external device address space (bit 31 of the address is set). Only the bus is frozen, so if the watchdog timer has been enabled, it will eventually reset the system.

Expected behavior is for such a write to cause a bus error interrupt. The bug does not affect the CPU interface *enhanced mode* (designed to be used with an external device such as Syd). In a system with an external device installed, SOLO1.3 behaves as expected.

Workaround: There are two workarounds.

- Ensure that the CPU never issues write transactions with bit 31 of the address set. This is guaranteed on a 4640-based system, since the CPU is not capable of generating such addresses. On a 5230- (or other CPU with an MMU) based system, the software should not access the external device's address space if the device is not present. This should not be a limitation.
- 2. Do not use compatibility mode even in a system with no external device. Tie off the external device control signals (pulldown on GPIO[2], pullup on TEST_SCANEN). This should behave exactly as workaround #1 except that if a write transaction is ever issued to the external device's address space, it will be completely ignored instead of halting the CPU bus.

Index

A	CelRecord 3-3, 3-4, 5-196, 6-72 celMode byte 5-206, 6-67, 6-78		
Alpha Blend 3-2, 5-191, 6-78 Alpha Blend modes Brighten 6-78 Force Background Opaque 6-78 Force Background Transparent 6-78 Alpha value 5-190, 5-191, 6-76 per-pixel Alpha (Aforegnd) 6-76 asynchronous devices DEV_CNTL registers 5-52 audUnit AUD_CLK 6-16 BITCLK 6-16 data out format AKM 4520 6-15 AKM 4532 6-15	full CelRecord 5-197 LoadCelsBase command 5-207 LoadCodeBook command 5-206 LoadInitColor command 5-207 LoadYMapBase command 5-206 microCelRecord 5-197, 6-76 miniCelRecord 5-197 SOLO1 State Machine 5-198 yMap 6-67 Chrominance 5-191 Closed Captioning 6-40 Codebook 3-9, 3-10, 3-11, 5-204 Color Look-Up Table 3-7		
Burr-Brown PM3001 6-15 SDATA 6-16 SDATAIN 6-16	data structures layers 5-194 devUnit		
В	Loop Mode bit 5-84 Output bit 5-84		
Brooktree 8-bit 'ByteStream' protocol 4-4 buffer chaining 6-17 Burst write transactions 6-21	Digital Video Encoder output modes Fixed IRE (FI) 4-4 Full Range (FR) 4-4 DIVIT port 4-2 divUnit		
<u>C</u>	windowing logic 6-26		
CCIR601 4-3, 6-40 CCIR656 4-4 CelBlocks 3-3, 5-197, 6-67	DMA chaining 5-60, 6-29 dveUnit RGB mode 6-40 YC mode 6-40		

E	memUnit Address Path and Arbiter block 6-43
EMV (Europay/MasterCard/Visa) 6-50, 6-53	arbitration Word Mode 6-46 auto-refresh 6-46 MEM_REFRESH register 6-46
Flash ROM 5-51 frame buffer vs. display formats 5-69	burp calculation 5-103 burp mechanism 6-46 Control Register block 6-43
<u>G</u>	- N
gfxUnit Frame Buffer Mode 5-188 GRFX Direct Mode 5-115 GRFX Write-back Mode 5-116	Non-Maskable Interrupt (NMI) 5-24
MEM_BURP register 5-118 ping-pong mode 5-113, 6-63 Write-back mode 5-119, 5-120, 6-63, 6-67 ACTIVELINES register 5-120	Overscan Region 6-83
Field mode 5-190 Frame mode 5-190 Write-back registers 5-117 yMap structure 5-194 topLine parameter 5-195 yMap_Ptr register 5-195 GRFX Direct Mode 5-189	Parallel Port Compatibility FIFO mode 5-86, 5-89, 6-34 ECP Reverse mode 5-86 Extended Capabilities Port (ECP) mode 5-86, 5-89, 6-34 hardware mode 5-86, 5-87 Software-handshake mode 5-89 software-only mode 5-86, 5-87 Philips 8002 interface 6-54
IDT R4640 6-86 IEEE1284 6-34 IR FIFO format 5-83	picture-in-picture (PIP) 4-1 POT_CLK 6-4 Q
ISO-7816 6-50, 6-53	QED RM5230 6-86
LRCLK 6-16 Luminance 5-191	Reset logic
M	Boot Mode register 6-90 Reset Cause register 6-90
Mask ROM 5-51 memory fences 4K granularity 5-25 cache-line granularity 5-25 FENCECNTL register 5-26	state machine 6-90 GONNARESET state 6-90 HOLDRESET state 6-90 NORMAL state 6-90 RESETTING state 6-90 Right/Left-Aligned Data Formats 6-15 rioUnit

ROM aliasing 5-49 BUS CHPCNTL register 5-49	V
ROM aliasing 5-49 BUS_CHPCNTL register 5-49 ROM_CNTL registers 5-50 S SAA7187 external DAC 6-36 Sasha ASIC 6-34 scanline algorithm 6-72 Scanline buffer 3-2, 3-9, 3-10, 3-12, 5-188, 5-190, 5-195, 6-76, 6-79 asynchronous single-ported memories 6-80 Screen coordinates 5-207 Security Access Module (SAM) 2-4 Smart Card Answer To Reset (ATR) 6-57 direct logic convention 6-61 elementary time units (ETUs) 6-59 inverse logic convention 6-61	Vector Quantization 2-6, 3-7, 5-192 formats 3-9, 3-10, 3-11 lossy compression 3-8 VQ4 Format 5-192, 5-204, 6-63, 6-76 VQ8 Format 5-192, 6-63, 6-75 video formats 720/768/640 2-7 JPEG 3-1 MPEG 3-1 NTSC, 3-1 PAL 3-1 YCbCr 3-1, 3-9, 5-191 vidUnit Chrominance averaging values DVEPIXAVG bit 5-77
terminal transport layer (TTL) 6-60 SMPTE 259M 6-26 sucUnit	VSYNCRESET bit 5-77 W
16550 UART 6-52 GPIO interrupt 5-181 GPU (general-purpose UART) 6-49 special-purpose output mode 5-178 WebTV Intermodule Interface (WTI) 6-50 synchronous devices WTV_CNTL register 5-55 T TextureMap 3-3, 3-5, 3-9, 3-10, 3-12, 5-208	watchdog timer counter 5-28 reset 5-19 RESETCAUSE_S register 5-28 WDREG_C register 5-28 WDVALUE register 5-28 WebTV Expansion Port 6-5 Expansion Port addresses 6-8 WebTV™ Memory Map 5-1
Lookup index 3-9 textureBase 3-6, 5-209 u and v parameters 5-208 vector cache 3-10 TextureMap formats 3-5 Turnaround Time 6-9	
U	
Underscan Region 6-83 Universal Asynchronous Receiver/Transmitter (UART) 6-49	